Министерство образования и науки РФ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Уральский государственный педагогический университет»
Институт математики, информатики и информационных технологий

ИННОВАЦИОННЫЕ ТЕХНОЛОГИИ В ПОДГОТОВКЕ УЧИТЕЛЕЙ ИНФОРМАТИКИ

Коллективная монография сотрудников кафедры информатики, информационных технологий и методики обучения информатике

Екатеринбург

2016

УДК 378.1
ББК Ч 448.44
И 66

Рецензенты

Н.В.Герова – доктор педагогических наук, профессор, зав.кафедрой информатизации и методики преподавания информатики Рязанского государственного университета

Л.В.Сардак – канд.пед.наук, доцент кафедры НИТО УрГПУ

ИННОВАЦИОННЫЕ ТЕХНОЛОГИИ В ПОДГОТОВКЕ УЧИТЕЛЕЙ ИНФОРМАТИКИ / коллективная монография / ФБГУ ВО «Уральский государственный педагогический университет» – Екатеринбург, 2016 – 120 с.

Представленная монография является коллективным трудом сотрудников кафедры информатики, информационных технологий и методики обучения информатике Института математики, информатики и информационных технологий УрГПУ. В ней представлены работы, отражающие в свете современных тенденций реформирования образования те научные направления, по которым кафедра ведет свою деятельность.

Для преподавателей, студентов, аспирантов, интересующихся вопросами обучения специальностям, связанным с информатикой.

УДК 378.1
ББК Ч 448.44

Содержание

ПРЕДИСЛОВИЕ

Попытки реформирования государственной системы образования предпринимаются уже четверть века. Дело в том, что советская система, несомненно, качественная могла существовать только в рамках автаркии, и в условиях рыночной экономики и глобализации она уже не эффективна. Поэтому не случайно, что в качестве фундамента реформы была выбрана Болонская система, дающая возможность интегрироваться в образовательное пространство Европы. Однако такой переход требует не только изменения механизмов управления, но и усовершенствования работы профессорско-преподавательского состава. А именно, методики преподавания должны быть нацелены на практическое применение изучаемых дисциплин, что в последних редакциях образовательных стандартов находит отражение в усвоении профессиональных компетенций. Кроме того, необходимо учитывать и то, что обучающийся должен сам формировать свою личность.

Представленная небольшая монография является коллективным трудом сотрудников кафедры информатики, информационных технологий и методики обучения информатике Института математики, информатики и информационных технологий УрГПУ. В ней представлены работы, отражающие в свете современных тенденций реформирования образования те научные направления, по которым кафедра ведет свою деятельность.

В первой главе, написанной доцентами кафедры А.М. Лозинской и И.В. Рожиной, рассмотрены направления профессиональной деятельности учителя школы в условиях информационной образовательной среды (ИОС) и компетенции, необходимые для их реализации, сгруппированные в кластеры: научно-теоретические, конструктивно-проектировочные, организационно-методические, профессионально-личностные. Предложены методические приемы развития кластеров профессиональных компетенций в процессе обуче-

ния учителей на основе: модульно-рейтинговой технологии с использованием фреймовых моделей когнитивной визуализации учебного материала. А также принципов обучения: «процесс важнее, чем результат» и «учение через преподавание»; видео-кейсов уроков, входящих в состав контента ИОС; активных методов обучения и технологии смешанного обучения.

Вторая глава, автор доцент А.И.Газейкина, посвящена вопросам обучения студентов программированию. Этот курс является одним из основных курсов предметной подготовки будущего учителя информатики и ИКТ. В процессе освоения содержания этой дисциплины у студента формируется представление о технологии создания программных средств, без которых невозможны современные информационно-коммуникационные технологии. Автор на основе своего богатого опыта определяет формы обучения в ходе лекционных занятий, а также методические приемы при проведении лабораторных работ с учетом индивидуальной коррекции. Выделены проблемные области при обучении студентов объектно-ориентированному программированию с одновременным использованием нескольких языков, как в рамках аудиторного обучения, так и при дистанционных формах.

В третьей главе, написанной профессором М.В.Лапенок, дан обзор современного состояние исследований в области создания и использования электронных образовательных ресурсов (ЭОС). Именно дистанционные образовательные технологии позволяют предоставить школьникам, студентам, гражданским и военным специалистам, безработным, самым широким кругам населения в любых районах страны и за ее рубежами равных образовательных возможностей. Автором дан обзор использования ЭОС и дистанционных методик обучения в России и за рубежом. Сформулированы принципы организации учебного процесса при дистанционных формах обучения. Выделены группы учащихся, для которых целесообразны именно эти формы обучения. А также определены организационные формы подготовки учителей, способных реализовать эти методики.

В четвертой главе (автор доцент Д.А.Емельянов) приведены общие подходы к созданию в среде визуальноного программирования Delphi практических приложений. В качестве примера дана разработка приложения “Шахматы”, в котором применение современной методики объектно-ориентированного программирования может быть проиллюстрировано наиболее ярко. Детально описаны все этапы создания такого приложения.

Пятая глава (автор П.И.Алексеевский) посвящена вопросам подготовки IT-специалистов к использованию современных технологий разработки программного обеспечения. Для обучения программированию с использованием методологии унифицированного процесса (УП) разработана методическая система, устанавливающая связь между фазами УП и этапами обучения.

В шестой главе (автор И.Е.Подчиненов) рассмотрены вопросы компьютерного моделирования сложных (или трудноформализуемых) объектов. Показано, что компьютерная модель может служить как средством теоретического исследования того или иного процесса, так и играть роль вычислительного эксперимента. Подчеркивается универсальность математических моделей, что позволяет их использовать для исследования различных процессов. Так в частности, модель кинетики химических реакций и модель взаимодействия двух видов «типа хищник-жертва» описываются по сути одной и той же математической моделью.

1. ФОРМИРОВАНИЕ И РАЗВИТИЕ ПРОФЕССИОНАЛЬНЫХ КОМПЕТЕНЦИЙ БУДУЩИХ ПЕДАГОГОВ С ИСПОЛЬЗОВАНИЕМ РЕСУРСОВ ИНФОРМАЦИОННО-ОБРАЗОВАТЕЛЬНОЙ СРЕДЫ

Профессиональное образование педагогов связано с формированием и развитием компетенций, которые являются не только многоплановыми и полифункциональными, но и динамично развивающимися. Анализ современного практического опыта профессиональной деятельности учителей показывает, что даже высокий уровень предметно-методической подготовки педагогических кадров не обеспечивает ожидаемых обществом результатов, достижение которых непосредственно связано с реализацией учебной деятельности с использованием инновационных образовательных технологий, методов, организационных форм и средств обучения. Учителя необходимо специально готовить к работе в информационно-образовательной среде (ИОС), принципиально новые дидактические возможности которой создают условия для реализации инновационных подходов к представлению содержания образования и организации учебного процесса. Таким образом, в современном информационном обществе главный акцент в профессиональном обучении педагогов следует делать на освоении продуктивных способов деятельности в ИОС в рамках реализуемого компетентностного подхода.

Проведем краткий функциональный анализ дидактических характеристик ИОС и составляющих профессиональной деятельности и компетенций учителя.

Вслед за Стариченко Б.Е. [6], определим ИОС как совокупность аппаратных средств, программных систем и контента, реализованную на основе современных технологических решений, и предназначенную для полного и опера-

тивного удовлетворения информационных потребностей всех субъектов учебного процесса, связанных с реализацией предусмотренных форм и видов учебной деятельности, а также организации информационных потоков, связанных с обучением и управлением учебным процессом.

Современная ИОС обладает рядом дидактических характеристик, к числу которых следует отнести: гибкость; целостность; открытость; вариативность; полифункциональность; интерактивность; мультимедийность; оперативность контроля учебных достижений; доступность разнообразных источников учебной информации; возможность организации индивидуальной работы обучающихся, развития их познавательной самостоятельности и творчества средствами информационно-коммуникационных технологий (ИКТ).

Реализация учебного процесса в ИОС, в отличие от традиционных условий, позволяет: расширить возможности выбора средств, форм и темпа изучения содержания обучения; обеспечить доступ к информационным ресурсам библиотек; слушать лекции ведущих учёных и задавать им вопросы; принимать участие в работе виртуальных школ; повысить интерес обучающихся к дисциплинам путем использования интерактивной, занимательной и наглядной формы представления учебного материала; актуализировать межпредметные связи; усилить мотивацию самостоятельного обучения и создать установку на непрерывное образование в течение жизни; активнее использовать методы взаимообучения (обсуждение учебных проблем на форумах, в чатах, оперативное получение подсказок); развивать инициативность, способность критически мыслить, ключевые компетенции учащихся.

Таким образом, именно ИОС придает учебному процессу качества, позволяющие достичь востребованных современным обществом образовательных результатов. При этом готовность учителей к эффективной работе в ИОС в значительной мере определяется пониманием необходимости существенной перестройки методической системы образовательного процесса. На первый план выходят профессиональные умения адаптировать в соответствии с дидактиче-

скими целями и конкретными педагогическими условиями (в том числе, связанными с ИОС образовательного учреждения) содержание обучения, выбрать оптимальные средства и методы реализации учебного процесса, обеспечить систематичный контроль учебных достижений и доступность его результатов для субъектов образовательного процесса, организовать коммуникационное взаимодействие с социальным окружением обучающихся и собственным административно-коллегиальным окружением.

Однако, как совершенно верно отмечается в результатах исследований ЮНЕСКО, современному учителю недостаточно быть технологически грамотным и уметь формировать соответствующие технологические умения и навыки у своих учеников: «современный учитель должен быть способен помочь учащимся использовать ИКТ для того, чтобы успешно сотрудничать, решать возникающие задачи, осваивать навыки учения и, в итоге, стать полноценными гражданами и работниками» [7].

Для эффективного, прогнозируемого формирования и развития требуемых профессиональных компетенций, связанных с решением задач обучения, воспитания и развития конкурентоспособной личности, следует опираться на характеристики профиля специалиста, определенные индикаторы поведения и использовать в качестве основы модель компетенций, которая должна отвечать следующим требованиям:

- соответствовать стратегическим целям образования;
- быть информативной и понятной для всех субъектов процесса образования;
- включать оптимальный набор компонентов;
- содержать систему учета и измерения компетенций.

Наиболее целесообразным инструментом формирования многомерных моделей (в нашем случае – модели профессиональных компетенций будущего педагога) выступает системный подход определения структуры.

Рассмотрим направления профессиональной деятельности учителя школы и компетенции, необходимые для их реализации.

1. Комплексная диагностика педагогических условий.

Компетенции, относящиеся к этой группе, позволяют учителю определить релевантные образовательные подходы для индивидуального обучения и использовать свои навыки наблюдения и базовые знания развития детей для определения сильных и слабых сторон образовательных стратегий в конкретных педагогических условиях. Для этого учитель должен уметь получить необходимую информацию о текущем функциональном состоянии учебного уровня учащегося (способности, познавательные потребности и личностные особенности, учебная мотивация), используя различные ресурсы среды, в том числе ИОС; уметь анализировать нормативную и учебно-методическую документацию; владеть методами сбора информации об учащемся и его окружении (тестирование, анкетирование, наблюдение, беседа и др.) и обладать определенными профессионально-личностным качествами в области коммуникации.

2. Разработка учебных целей и задач.

Навыки в этой группе включают в себя разработку обучающих целей в терминах наблюдаемых результатов деятельности и/или поведения, на основании которых будут оцениваться учебные результаты. Цели должны формулироваться в ясных и недвусмысленных терминах деятельности, позволяющих точно поставить задачи, а затем проверить успешность и эффективность их выполнения.

3. Анализ учебных задач.

В рамках сформулированных целей и поставленных задач учитель должен уметь определить необходимую учебную деятельность и возможные адекватные методики обучения. Для этого необходимо уметь: выделить в учебных задачах значимые составляющие и соотнести их с деятельностным аспектом реализации содержания обучения; провести анализ современных средств реше-

ния поставленных задач (педагогических, методических, технико-технологических, программных); методически грамотно определить адекватную последовательность обучения.

4. Выбор, методическая коррекция и использование учебных материалов.

Учитель должен иметь представление о широком спектре учебных средств массовой информации, учебно-методических материалов и критериев их отбора, а также способах модификации. Кроме того, необходимо уметь привлекать и адаптировать к учебному процессу материалы из других образовательных организаций и смежных областей знаний и деятельности. Современный учитель также должен уметь использовать и разрабатывать электронные образовательные ресурсы, обладать культурой работы с информацией в сети интернет.

5. Выбор, методическая коррекция и использование технологий и стратегий обучения.

Компетенции, необходимые для выбора и использования соответствующих учебных стратегий требуют осведомленности учителя о различных учебных процедурах, методических приемах и технологиях, доступных для дидактической инженерии и эффективного управления процессом обучения, а также обладания умением адекватно соотнести цели и возможности обучающихся с этими стратегиями для принятия наиболее оптимального решения.

6. Контроль и оценка результатов образования.

К этой группе компетенций относятся навыки управления учебным процессом, конструирования контрольно-измерительных материалов на основе разработанных учебных планов, разработки и применения методик контроля и оценки, в том числе и с использованием информационно-коммуникационных технологий и информационно-образовательной среды. Компетенции, связанные с оценкой достижений учащегося требуют мастерства в эмпирическом исследо-

вании, что в определенной степени обусловлено экспериментальной природой диагностики обучения.

7. Использование ресурсов.

Знания и умения данной группы профессиональных характеристик связаны с поиском, отбором, модификацией и использованием учебных материалов и технологий обучения, причем мастерство в использовании информационно-поисковых систем и ИОС является очень важным фактором эффективного использования времени и достижения конечного результата. Компетенции, связанные с умением использовать ресурсы (материальные, информационные, технико-технологические, людские и другие) и поддерживать благоприятные отношения с ними, позволяют учителю разрабатывать и использовать специализированные учебные курсы, уникальные технологии обучения, средства обучения и программные среды; предлагать и получать образовательные и сопутствующие им услуги с привлечением соответствующих ресурсов.

8. Управление поведением.

К группе характеристик управления поведением относится владение способами укрепления желаемого поведения, формирования новых моделей поведения и замены ими проявлений нежелательных форм поведения. Учитель должен владеть методами модификации поведения, установления приемлемых границ поведенческих реакций, использования развивающих приемов, направленных на формирование конструктивной модели поведения. Безусловно, учитель должен уметь не только применять навыки управления поведением в соответствии с технологическим требованием учебного процесса, но и демонстрировать их в каждодневной работе без специально поставленных целей. Отметим, что учитель проявляет компетентность взаимодействия с обучающимися в трех направлениях: стимулирование желаемого взаимодействия; укрепление желаемого взаимодействия; соответствующее реагирование на нежелательное взаимодействие.

9. Профессиональная деятельность.

Компетенции, связанные с профессиональной деятельностью основываются на признании необходимости постоянного индивидуального самосовершенствования, профессионального развития и обновления, обучения на протяжении жизни, участия в деятельности профессиональных организаций, и использования накопленного практического опыта и баз знаний в области образования для проведения эмпирических исследований. Учитель проявляет компетентность в области профессиональной деятельности, участвуя в работе органов законодательства в области образования, конгрессов, конференций, семинаров; анализируя информацию из профессиональных журналов; осваивая новые дидактические методы и педагогические приемы; обмениваясь накопленным педагогическим мастерством и идеями с другими. Особенно важны умения учителя развиваться в профессиональном плане, с использованием возможности сети интернет и информационно-коммуникационных и телекоммуникационных технологий.

10. Знание современных тенденций.

Учитель должен обладать не только знаниями и навыками в своей области образования, но и знать много контекста и направлений, в которых они могут быть применены. Обладая компетенциями в этой области, учитель более правильно оценивает свои сильные и слабые стороны, а также свою роль в развитии профессиональной сферы деятельности и общества. Для этих компетенций также важно иметь представление о новых информационно-коммуникационных технологиях, технических достижениях.

11. Предметная (специальная) обученность.

Безусловно, компетентный учитель хорошо знает то, чему он учит других. Содержание компетенций этой группы охватывает когнитивную, аффективную и психомоторную области знаний.

12. Взаимодействие с родителями и ближайшим окружением учащихся.

Не вызывает сомнения, что участие родителей очень важно в содействии развитию своих детей, родительские отношения и ожидания влияют на будущее учащихся. Компетенции в консультировании родителей требуют навыков в межличностных отношениях и знаний о динамике человеческих взаимодействий.

В соответствии с выявленной структурой педагогической деятельности, в работе учителя в условиях ИОС можно выделить следующие компоненты:

(a) гностический – изучение и анализ учителем различных видов деятельности обучающихся при организации учебного процесса в ИОС, а также возможностей сервисов системы для создания и использования электронных образовательных ресурсов; выявление педагогически целесообразных форм и методов обучения в ИОС с учетом целей занятия, форм информационного взаимодействия обучающего с обучаемыми, знаний и интересов обучающихся, технологических возможностей системы обучения; получение и анализ информации о состоянии учебного процесса с использованием сервисов ИОС;

(b) проектировочный – определение дидактических целей и задач учебного процесса, организованного в ИОС; планирование структуры и формы представления содержания обучения с целью поддержки всех видов учебных занятий в соответствии с образовательной программой, а также методов обучения, стимулирующих исследовательскую деятельность обучающихся;

(c) конструктивный – разработка и оперативная коррекция планов занятий (в том числе индивидуальных учебных планов обучающихся), учебно-методических материалов, электронных образовательных ресурсов.

(d) организаторский – ведение учебных занятий с использованием ИОС; и электронного документооборота; инструктирование учащихся по работе с сервисами информационной среды; организация индивидуальной и групповой исследовательской деятельности учащихся в условиях удаленного и непосредственного информационного взаимодействия; управление контролем результа-

тов обучения.

(е) коммуникативный – осуществление непосредственного и дистанционного информационного взаимодействия между участниками учебного процесса в ИОС; общение между учителями для расширения профессиональных контактов с использованием сервисов информационной среды.

Таким образом, нами выявлено, что компетенции будущего учителя связаны с его профессиональной реализацией и развитием, а также функционированием во взаимодействии с социальной, технико-технологической и организационно-структурной системами.

Напомним, что компетенции включают в себя такие элементы как знания, умения и навыки, а также способности и потенциальные возможности, относящиеся к деловым и личностным качествам (рис. 1). Одновременно они являются привычными шаблонами мышления и поведения, использование которых делает человека успешным в конкретной работе или роли [2].

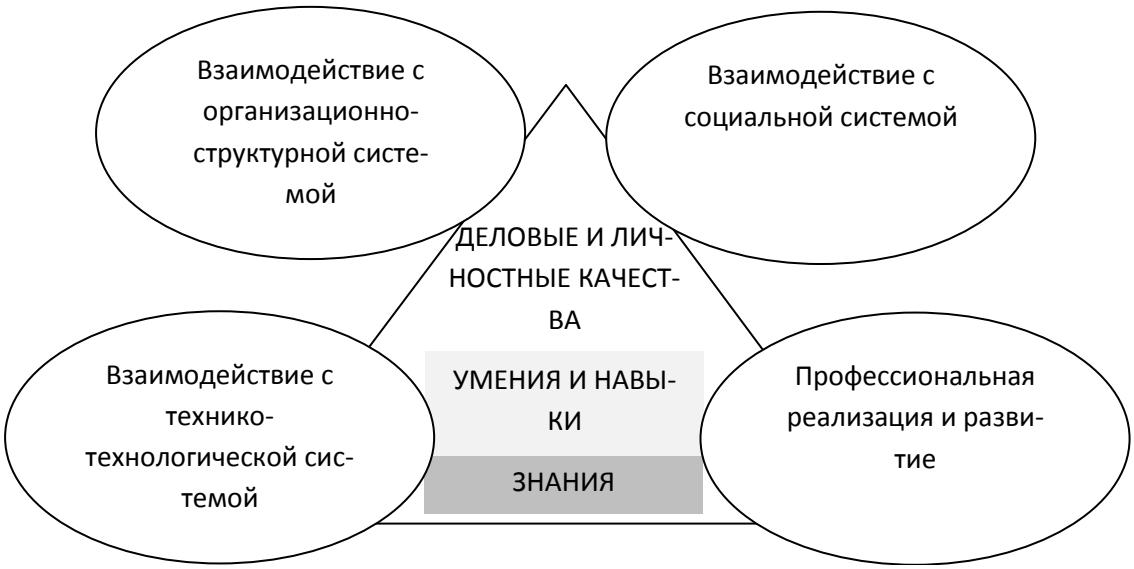

Рис.1. Области проявления профессиональных компетенций учителя

Для формулирования обобщенных требований, компетенции можно объединить в кластеры – наборы компетенций (обычно 3-5), тесно связанных между собой родственными индикаторами характеристик. В рамках нашего иссле-

дования были определены основные кластеры компетенций в соответствии с функциональным содержанием профессиональной деятельности учителя: научно-теоретические; конструктивно-проектировочные; организационно-методические; профессионально-личностные.

Проанализируем кратко составляющие выделенных кластеров профессиональных компетенций в условиях ИОС, задачи и проблемы их формирования.

Научно-теоретические компетенции: специальная обученность; сформированность междисциплинарных представлений; сформированность научного мировоззрения.

Одним из факторов, определяющих качество образования, является содержание специальных (предметных) компетенций учителя. Они представляют собой педагогическую адаптированную систему: научных знаний; способов деятельности (умения действовать по образцу); опыта творческой деятельности; опыта эмоционально-ценностного отношения к природе, обществу и человеку. Очевидно, что составляющие профессионально-педагогической компетентности учителей различных учебных дисциплин имеют определенные доминанты, что обусловлено спецификой предмета и методикой его преподавания.

Эти компетенции проявляются в информированности учителя об основных идеях, понятиях, концепциях в предметных областях знаний; сформированности общеучебных (дидактических) умений и навыков, интеллектуальных способностей в самостоятельном приобретении новых знаний, средств и способов познавательной деятельности и другом. Они обеспечивают навыки деятельности с информацией, содержащейся в учебных предметах и образовательных областях, а также в окружающем мире.

Учитель демонстрирует наличие потребности к познавательной деятельности; умение ориентироваться и пользоваться различными информационными

источниками для получения новых знаний; владение дидактическими умениями и навыками образовательной деятельности; сформированное целостное представление о картине мира, выбор собственной мировоззренческой позиции; умение выявить закономерности в основе изучаемых наук, норм, правил общественной жизни.

Конструктивно-проектировочные компетенции: навыки педагогического прогнозирования; навыки разработки современных систем обучения (инноваций); навыки разработки учебно-методических материалов для организации обучения.

Проектировочные компетенции предполагают владение теоретическими методами действий при разработке целостного процесса и учебных занятий на основе прогрессивных педагогических технологий.

Ключевую роль в профессиональной деятельности современного педагога играют умения проектирования учебного процесса в ИОС, как целостного, отражающего взаимосвязь всех компонентов (урочная, исследовательская деятельность, измерение, контроль и оценка результатов обучения). ИОС имеет свои дидактические возможности, которых раньше не было в арсенале учителя (гибкость, адаптивность, вариативность среды, её трансформируемость из одной «версии» в другую, настраиваемость под решение различных учебных задач). Переход учителя к работе в ИОС предполагает изучение и анализ возможностей, методов, форм и средств обучения, характерных для этой среды, а также видов учебной деятельности школьников, обеспечивающих достижение новых образовательных результатов.

Организационно-методические компетенции: готовность осуществлять педагогическую деятельность; готовность применять современные методики и технологии обучения; умения работать в коллективе и с коллективом.

В условиях осуществления учебного процесса в ИОС изменяется характер взаимодействия его участников: учащиеся выступают в роли субъекта дея-

тельности, в отличие от традиционной образовательной среды, где он выполнял роль объекта; роль учителя и содержание его профессиональной деятельности (по гностическому, организационному, проектировочному, экспертному, рефлексивному компонентам).

Содержание подготовки учителя к проектированию учебного процесса в ИОС должно включать обязательные элементы: понятие информационной образовательной среды в условиях модернизации педагогического образования; компонентный состав учебного процесса в ИОС; технология проектирования учебного процесса в ИОС.

Профессионально-личностные компетенции: навыки коммуникации; навыки управления; готовность к саморазвитию; владение профессиональной этикой.

Анализ исследований в области подготовки учителей в ИОС показал, что современный педагог должен обладать коммуникативной компетенцией, то есть совокупностью знаний, умений и личностных качеств, позволяющих строить эффективное взаимодействие в электронной среде с другими субъектами, непосредственно участвующими в педагогическом процессе.

Компетентность педагога в области управления в условиях ИОС основывается на знаниях в области менеджмента; умениях осуществлять опережающее планирование, моделирование и прогнозирование процесса обучения, эффективно использовать информационные ресурсы; навыках управления собственной деятельностью и деятельностью учащихся.

Компетентный учитель должен соблюдать правовые, нравственные и профессионально-этические нормы: проявлять корректность и внимательность к обучающимся, терпимость и уважение к обычаям и традициям народов, учитывать особенности психофизического развития обучающихся и состояние их здоровья, способствовать формированию благоприятного морально-психологического климата для эффективной работы, своим личным поведени-

ем подавать пример честности, беспристрастности и справедливости, а также создавать благоприятное впечатление самопрезентацией личностных качеств во внешнем облике – стилем одежды, общей аккуратностью. Большое влияние на качество реализации профессиональных функций учителя оказывает проявление личностных качеств в системе социальных отношений: толерантность, эмпатия, социальная мобильность, целость и идентичность личности, ее готовность к самовоспитанию.

В процессе подготовки студентов в педагогическом вузе формируется профессиональная компетентность, которая представляет собой интегральную характеристику, отражающую мотивационно-ценностные ориентации личности, наличие способностей решать профессиональные проблемы и задачи, возникающие в реальных ситуациях педагогической деятельности, с использованием знаний в различных сферах и профессиональной области, профессионального и жизненного опыта, умением осуществлять рефлексию педагогической деятельности и самосовершенствование.

Рассмотрим модель формирования и развития профессиональной компетентности студентов педагогических вузов. Выстроенная модель имеет поэтапную структуру, в которой на каждом из этапов реализуется часть методической системы профессионального образования (рис. 2).

Рассмотрим методические приемы развития кластеров профессиональных компетенций будущих педагогов в условиях ИОС.

1. Развитие научно-теоретических компетенций.

В целях повышения эффективности учебного процесса, развития логического и технического мышления следует: 1) Использовать методы проблемного обучения, способствующие осознанному усвоению знаний. Высоко результативной показала себя методика М.А. Чошанова, основанная на реализации подходов «процесс важнее, чем результат», «учение через преподавание» и «учение через анализ и рефлексию». Методика включает в себя систему учебных

работ с видео-кейсами уроков: задание-решение до просмотра; задание-пауза во время просмотра; задание-рефлексия после просмотра [3]. 2) Разрабатывать содержание образования с учетом достижений современной нейропедагогики и инженерии знаний. Использовать различные модели фреймового структурирования и представления информации, учитывая, что когнитивные цели освоения содержания образования определяют использование в процессе конструирования учебного материала следующих моделей фреймов: фрейм-рамка; фрейм-логико-смысловая схема; фрейм-сценарий [4]. Фреймы в виде рамки и логико-смысловой схемы целесообразно применять для визуализации семантической структуры учебного материала. Фреймы в виде сценариев позволяют эффективно обучать типовым алгоритмам деятельности для решения разноплановых педагогических задач.

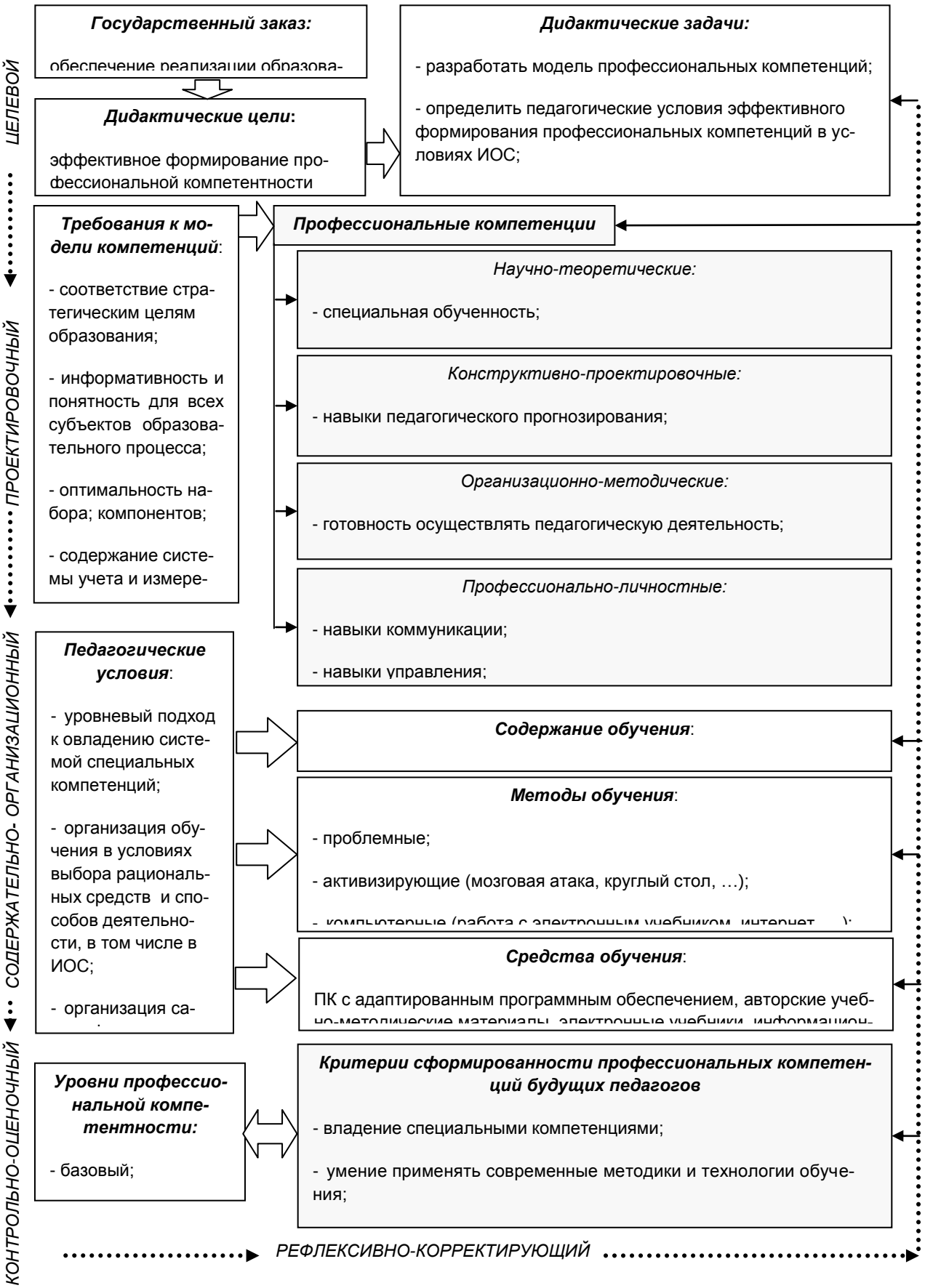

Рис. 2. Модель формирования и развития профессиональных компетенций будущих педагогов

2. Развитие конструктивно-проектировочных компетенций.

При подготовке будущего учителя к проектированию учебного процесса в ИОС, надо учитывать, что образовательный процесс: а) следует строить так, чтобы обучаемый имел возможность открыть для себя личностно значимый смысл в формировании профессиональных компетенций, необходимых ему для работы в ИОС; б) должен осуществляться в деятельностной форме и на основе дифференцированного подхода к обучаемым, создания условий для построения индивидуальной траектории обучения в соответствии с личностными потребностями и возможностями; с) должен обеспечивать условия для проявления самостоятельности и инициативности обучающихся, способностей работы с современными педагогическими инструментами и актуализации разнообразных рефлексивных процедур; д) может быть построен в условиях сетевого взаимодействия образовательных учреждений, располагающих различными методическими, информационными и кадровыми ресурсами, необходимыми для эффективной подготовки учителя к проектированию учебного процесса в ИОС.

В исследовании Лапенок М.В. и Рожиной И.В. [1] определена структура и содержание подготовки учителей к созданию и использованию в учебном процессе школы, в условиях ИОС, электронных образовательных ресурсов (ЭОР). В соответствии с разработанными ими научно-методическими подходами, в процессе обучения будущих учителей следует: создавать образовательные ресурсы с учетом дидактических возможностей инструментальных сервисов ИОС и дидактических, эргономических и инновационных требований к ЭОР; применять активно-деятельностные формы обучения – ролевые игры, которые моделируют типовые педагогические ситуации, присущие учебно-познавательному процессу с использованием ЭОР, и способствуют формированию опыта учителей по интерактивному информационному взаимодействию в режимах реального времени и отложенной связи с использованием инструментальных сервисов ИОС.

3. Развитие организационно-методических компетенций.

В процессе обучения будущих педагогов должны использоваться разнообразные инновационные подходы: обучение на основе информационных ресурсов, посредством телеконференций; технологии обучения в сотрудничестве (проекты, форумы, электронные семинары), игровые технологии; методы ассоциаций, искусственного интеллекта, «вынужденных предположений», прецедента, реификации; учебное компьютерное моделирование, эксперимент, научно-методическое исследование и др..

Система оценки учебных достижений также может осуществляться посредством активного участия студентов в учебных дискуссиях по основным темам курса, представленным в ИОС; составлении аналитических обзоров по информационным ресурсам; рефлексии по видео-кейсам школьных уроков.

4. Развитие профессионально-личностных компетенций.

Использование активных методов обучения способствует не только включению студентов в организацию педагогического процесса и формированию управленческих компетенций, но и развитию коммуникативных навыков, освоению норм профессиональной этики.

Одним из методов, который позволяет поддерживать студента в достижении учебных целей и поощрять на достижение новых результатов, фиксировать динамику роста за определенный период времени, презентовать спектр и качество выполненных работ, является *метод портфолио.* Включение *метода мозговой атаки и дебатов* в обучение при реализации данного кластера позволяет сформировать умения формулировать и отстаивать собственную точку зрения, делать выводы, выстраивать цепочку доказательств, выявлять ошибки, анализировать полученную информацию, концентрироваться на сути проблемы, работать в команде. Общепризнан высокий потенциал *метода проектов и исследовательского метода* в вопросах повышения познавательной активности и творческих способностей, без которых невозможно осознанное восприятие материала.

Отметим некоторые общие вопросы построения методической системы, направленной на повышение уровня профессиональных компетенций будущих педагогов в условиях ИОС. Одним из наиболее перспективных направлений повышения эффективности процесса профессионального образования является использование в учебном процессе модульных технологий, характеризующихся высоким уровнем достижения запланированных результатов обучения и их воспроизводимостью, а также структурной, технологической и содержательной гибкостью модульных программ. Эффективность модульной технологии во многом обусловливается быстрым развитием электронных образовательных ресурсов, реализацией компетентностной парадигмы образования, применением рейтинговой системы контроля качества обучения [5].

Для повышения уровня профессиональных компетенций при организации учебного процесса представляется целесообразным использование *технологии смешанного обучения* (blended learning), поскольку она обладает достоинствами distance learning и компенсирует его недостатки. В процессе реализации смешанного обучения применяются различные событийно-ориентированные методики и схемы управления учебным процессом, такие как *face-to-face learning*, *distance learning* и *on-line learning*. При этом обучение строится на взаимодействии обучаемого не только с компьютером, но и с преподавателем в активной форме (очной и дистанционной), когда изученный самостоятельно материал обобщается, анализируется и используется для решения поставленных задач. В основе предлагаемого нами подхода к организации смешанного обучения лежит дистанционный курс, в который интегрированы методы активного обучения, реализующиеся на очных занятиях со студентами, и основанные на сочетании групповых и индивидуальных, реальных и виртуальных форм, а также целенаправленной, интенсивной и контролируемой самостоятельной работе студентов.

Учебный процесс в схемах смешанного обучения может строиться на основании следующих положений [6]: 1) часть теоретического учебного материа-

ла студенты изучают самостоятельно *(асинхронный off-line режим)* до проведения аудиторной лекции (в сроки, установленные графиком изучения дисциплины), что позволяет преподавателю читать лекцию подготовленной аудитории и сместить акценты с простого пересказа материала на совместное обсуждение и разбор моментов, вызвавших затруднение студентов; 2) часть аудиторных занятий переводится в интерактивный дистанционный режим *(синхронный on-line)*, что увеличивает охват участников и позволяет проводить занятия параллельно для многих студентов, удаленных от преподавателя; 3) проведение части семинаров и практических занятий также возможно в on-line режиме посредством неинтерактивных сервисов (*форумы, чаты*) или в интерактивной режиме (*виртуальная классная комната*); 4) контроль и самоконтроль усвоения материала осуществляется посредством тестовой системы, которая является сервисом ИОС; 5) управление процессом обучения производится через неинтерактивные коммуникационные сервисы.

Гибкость модели смешанного обучения состоит в том, что пропорции «смешивания» традиционной очной и дистанционной форм могут быть различными.

На основе анализа подходов, изложенных в педагогической и научной литературе, можно сделать вывод о том, что повышению уровня профессиональной компетентности способствует применение определённых методов, приёмов и форм деятельности. Среди форм организации обучения приоритет отдаётся тем, посредством которых возможно организовать активную работу над изученным материалом и с его применением.

В рамках нашего исследования было установлено, что *для диагностики сформированности кластеров профессиональных компетенций будущих педагогов* может быть успешно применена методика, предложенная для измерения и оценки уровня сформированности компетенции в области использования инструментальных сервисов ИСДО [1], в которой были выделены когнитивно-операционная (знания, умения) и деятельностная (опыт) компоненты компетен-

ции, разработаны шкалы оценивания уровней сформированности каждой из компонент и методика расчета интегрального показателя уровня сформированности компетенции.

Обобщая изложенное выше, можно сделать вывод, что в эпоху развитых цифровых технологий обществу нужны учителя, в компетенции которых интегрированы знание содержания обучения, педагогической психологии, ресурсов ИКТ и навыки дидактического проектирования. Интеграция также предполагает изменение роли педагога: новые образовательные результаты могут быть достигнуты только в процессе освоения современных видов учебной деятельности, то есть в инновационном образовательном процессе, выстроенном в ИОС, что с неизбежностью влечет за собой смену традиционного учения научно-исследовательской техникой обучения студентов.

Эти преобразования требуют развития у учителя следующих умений: 1) разрабатывать учебные цели, ориентированные на достижение результатов в технологически расширенной информационной среде, которые позволяют студентам устанавливать свои собственные цели обучения, контроля и оценки прогресса в учении; 2) проектировать содержание образования в виде интерактивного контента и соответствующих практических упражнений путем выбора и проектирования задач, проектов и мероприятий с использованием цифровых ресурсов и ИКТ для формирования учебного опыта и развития исследовательских, проектировочных и творческих умений студентов; 3) разрабатывать систему контроля и оценки в соответствии с целями обучения и содержанием для комплексной объективной диагностики учебных результатов, улучшения качества преподавания и мотивации студентов к учению.

Повышению уровня профессиональной компетентности способствует применение активных методов, приёмов и форм учебной деятельности: проблемных, эвристических, имитационно-игровых, моделирующих, проективных, дискуссионных и др. Применение активных методов обучения способствует переносу знаний и умений студентов в новую ситуацию, изучению новых про-

блем, формированию умений видеть альтернативу решения, комбинировать известные способы решения и создавать новые, оригинальные алгоритмы деятельности.

Развитие педагогической науки в направлении использования электронных дидактических сред открывает новые возможности для понимания задач и форм образования в цифровую эпоху, создания эффективных методов обучения.

ЛИТЕРАТУРА

1. Lapenok M.V., Rozhina I.V. Teachers' training and comprehensive assessment of their educability level in the development and use of electronic educational resources. In The collection International scientific-practical conference Innovations in science, technology and the integration of knowledge, P.113-122. Berforts Information Press, London, 2014.
2. School Turnarounds Teachers: Competencies for Success (2008). Public Impact for the Center for Comprehensive School Reform and Improvement. URL: http://publicimpact.com/publications/Turnaround_Teacher_Competencies.pdf (дата обращения 13.12.2015).
3. Tchoshanov M.A. Engineering of learning: Conceptualizing e-Didactics. 2013. Published by the UNESCO Institute for Information Technologies in Education. URL: http://iite.unesco.org/pics/publications/en/files/3214730.pdf (дата обращения 13.12.2015).
4. Лозинская А.М. Фреймовое структурирование содержания обучения физике в рамках модульной технологии // Педагогическое образование в России. – 2014. – № 1. – С. 80-89.
5. Лозинская А.М., Шамало Т.Н. Модульное структурирование содержания обучения дисциплине // Педагогическое образование в России. – 2014. – № 3. – С. 39-44.

6. Стариченко Б.Е. Концептуальные основы компьютерной дидактики. Учебное пособие / Урал. гос. пед. ун-т. Екатеринбург, 2013. – 139 с.
7. Структура ИКТ-компетентности учителей. Рекомендации ЮНЕСКО. URL: http://iite.unesco.org/pics/publications/ru/files/3214694.pdf (дата обращения 13.12.2015).
8. Тырина М.П. Дидактическая компетентность педагога и ее развитие / Известия алтайского государственного университета. – 2012. – № 2-1. URL: http://izvestia.asu.ru/2012/2-1/peda/TheNewsOfASU-2012-2-1-peda-07.pdf (дата обращения 28.06.2015).
9. Федеральный закон "Об образовании в Российской Федерации" URL: http://www.assessor.ru/zakon/273-fz-zakon-ob-obrazovanii-2013/ (дата обращения 28.06.2015).
10. Rozhina I.V., Lozinskaya A.M., Shamalo T.N. Raising the Level of Future Teachers' Professional Competence in the Conditions of Informational and Educational Environment. In The collection International scientific-practical conference Smart Education and Smart e-Learning. Smart Innovation. Systems and Technologies. Vol 41 Series editors V.L.Uskov,R.J.Howlett, L.C.Jain. Springer International Publishing. Switzerland. 2015. DOI 10.1007/978-3-319-19875-0_29

2. ОБУЧЕНИЕ ЯЗЫКАМ И ТЕХНОЛОГИЯМ ПРОГРАММИРОВАНИЯ КАК КОМПОНЕНТ ПРЕДМЕТНОЙ ПОДГОТОВКИ БУДУЩЕГО УЧИТЕЛЯ ИНФОРМАТИКИ

Значение предметной подготовки будущего учителя информатики невозможно переоценить: в школу к детям должен прийти учитель, не только хорошо знающий педагогику, психологию и методику обучения, но и являющийся специалистом в области информатики и информационно-коммуникационных технологий, обладающий предметными компетенциями, знаниями и умениями, выходящими далеко за пределы школьного учебника.

Если образовательный стандарт второго поколения выделял цикл дисциплин предметной подготовки будущего учителя (и регламентировал их содержание), то ФГОС третьего поколения по направлению «Педагогическое образование» унифицировал содержание подготовки будущего учителя в области гуманитарных, естественнонаучных и математических, а также общепрофессиональных дисциплин, предоставив разработчикам образовательной программы спроектировать предметную составляющую подготовки в виде дисциплин вариативной части [16]. Эти дисциплины предметной подготовки будущего учителя информатики должны охватить широкий спектр различных разделов информатики: от программирования и робототехники до архитектуры компьютера и компьютерных сетей.

Курс программирования является одним из основных курсов предметной подготовки будущего учителя информатики и ИКТ. В процессе освоения содержания этой дисциплины у студента формируется представление о технологии создания программных средств, без которых невозможны современные информационно-коммуникационные технологии. Курс формирует у студентов представление о способах разработки программных средств современных информационных технологий, учит разрабатывать эти средства на основе использования современных языков, методов и технологий программирования. Поэтому содержание, формы и методы обучения программированию должны со-

ответствовать современному состоянию языков, методов и технологий программирования и перспективам их развития.

Однако перед преподавателем, читающим данный курс, встает ряд проблем, решение которых необходимо для эффективной подготовки будущего учителя информатики.

Специфика педагогического вуза такова, что в число первокурсников, как правило, не попадают вчерашние школьники, имеющие высокий уровень знаний и умений в области программирования. Анализ анкет и результатов входных тестов по программированию студентов первого курса Института математики, информатики и информационных технологий Уральского государственного педагогического университета показывает, что в течение шести последних лет (период, когда итоговая аттестация в школе проводилась в форме ЕГЭ, и при зачислении абитуриентов учитывался балл, полученный по информатике и ИКТ) около 40% студентов первого курса в школе не изучали программирование ни в каком его виде.

В качестве входного теста тем первокурсникам, которые занимались программированием в школе, было предложено составить три программы на любом языке программирования (либо алгоритм в виде словесного описания или блок-схемы): осуществить поиск большего из двух чисел; вычислить сумму входящей числовой последовательности; упорядочить элементы массива по возрастанию/убыванию. В результате выяснилось, что десятая часть выпускников школы помнит только отдельные ключевые слова и команды языка и не в состоянии составить даже простейшей программы (алгоритма). Около 50% первокурсников могут выполнить лишь первое задание на составление разветвляющейся программы; примерно 30% знают, что такое цикл и справились со вторым заданием, и лишь единицы правильно обрабатывают массив и могут реализовать алгоритм его сортировки.

Примерно четвертая часть первокурсников, изучавших в школе программирование, осваивала язык Basic, большая часть обучалась программированию на языке Pascal, отдельные студенты имеют представление о программирова-

нии на C/C++. По результатам опроса около 80% первокурсников определили свой уровень подготовки по программированию как нулевой или низкий, остальные – как средний, и никто не оценил как высокий.

Таким образом, уровень подготовки первокурсников в области программирования является крайне низким. Это то, что мы имеем «на входе». «На выходе», т.е., по прошествии положенного срока обучения в вузе, мы должны получить выпускника, который, в соответствии с требованиями образовательного стандарта [16], овладеет знаниями, умениями и практическим опытом в области объектно-ориентированного, логического и web-программирования. Не рассматривая двух последних пунктов в этом перечне, остановимся более подробно на обучении будущих учителей информатики «классическому» программированию.

При этом можно обозначить ряд проблем:

– выбор начального языка программирования и языков для последующего обучения;

– отбор содержания обучения;

– выбор методов, форм и средств обучения, в том числе основанных на применении информационно-коммуникационных технологий;

– реализация межпредметных и внутрипредметных связей т.п.

Традиционно, обучение программированию будущего учителя информатики включает в себя освоение студентами сначала структурного программирования (Pascal), а затем объектно-ориентированного (Object Pascal, Java, C++). Как показывает анализ статей и публикаций по этой проблеме, большая часть педагогических вузов придерживается такого же подхода при обучении своих студентов.

В реализуемом подходе не вполне понятным остается назначение первого раздела, т.е. обучение студентов структурному программированию. Основными доводами, приводимыми в качестве обоснования существования этого раздела в вузовском курсе программирования, являются невозможность изучения объ-

ектно-ориентированного программирования без опоры на знания и умения в области структурного, а также тот факт, что осуществляется процесс подготовки будущего учителя информатики, который, придя затем в школу, должен будет обучать школьников структурному программированию (и, как правило, на языке Pascal).

Однако, еще Э. Дейкстра в своей книге «Дисциплина программирования» отмечал, что «наиболее незаметным свойством любого инструмента является его влияние на формирование привычек людей… . Когда этот инструмент – язык программирования, его влияние, независимо от нашего желания, сказывается на нашем способе мышления» [9, с. 4]. Поэтому проблема выбора языков программирования, отбора содержания и выбора адекватных методов и средств обучения в процессе профессиональной подготовки будущего учителя информатики в настоящее время весьма актуальна. Сегодняшние студенты начнут свою профессиональную деятельность через 3-4 года после изучения курса программирования в вузе. За это время индустрия программирования продвинется вперед в развитии языков, методов и технологий программирования. Поэтому нецелесообразно обучать тому, что уже сегодня является вчерашним днем в области информационных технологий. Структурный подход к разработке программ должен быть заменен объектно-ориентированным [5, 6, 14], являющимся в настоящее время основной парадигмой, используемой в индустрии программирования. Он является основным и при программировании для Интернет (разработка веб-сервисов, создание аплетов, создание динамического контента).

Таким образом, первым языком программирования для студентов педагогического вузы должен быть объектно-ориентированный язык. В качестве него может выступать Object Pascal (с его реализацией в визуальной среде программирования Delphi) – это облегчит усвоение материала теми студентами, которые изучали структурное программирование на Pascal в школьном курсе информатики. Полновесной заменой Delphi может служить свободная среда разработки Lazarus.

Другой вариант выбора первого языка программирования в вузе – это Java, который является полностью объектно-ориентированным языком, позволяющим разрабатывать платформо-независимые приложения, а также аплеты и приложения для мобильных устройств. Для разработки Java-программ можно использовать интегрированную среду разработки Eclipse (это среда с открытым исходным кодом, для ее использования не требуется покупка лицензии) либо интегрированную визуальную среду NetBeans, которая поставляется вместе с Java 2 SDK Standart Edition последних версий. Третий вариант выбора – язык C++ как наиболее распространенный профессиональный язык программирования для разработки приложений самого широкого назначения.

Представляет интерес подход к обучению программированию студентов, основанный на одновременном (параллельном) изучении и использовании сразу двух языков программирования. В 2010-2011 учебном году такой эксперимент был начат со студентами первого курса Института математики, информатики и информационных технологий УрГПУ, он продолжается до настоящего времени. Для обучения программированию студентов первого курса были выбраны Object Pascal и Java, на третьем курсе к этим двум языкам добавляется сначала язык C, а затем C++. Достоинствами такого подхода являются:

– возможность использования разного инструментария для решения одной и той же задачи;

– возможность сравнительного анализа возможностей разных языков программирования и формирование способности выбирать оптимальный инструментарий для решения поставленной задачи;

– развитие алгоритмического и логического мышления студентов.

Еще одной проблемой для преподавателя является обоснованный выбор способов обучения (т.е. методов, организационных форм и средств обучения). Особое значение эта проблема приобретает при обучении именно будущего учителя, поскольку [4, 13, 15] он затем в свой профессиональной деятельности транслирует применявшиеся к нему формы, методы и приемы обучения. В настоящее время неприемлемой (и это отмечено в Федеральных государственных

образовательных стандартах [16]) является традиционная форма организации обучения «лекции – лабораторные занятия», при которой в ходе лекции рассматриваются возможности языка программирования, а на лабораторных занятиях студенты выполняют задания на составление программ с использованием рассмотренных на лекции возможностей языка программирования.

Опыт показал, что при организации изучения теоретического материала в ходе лекционных занятий целесообразно использовать следующие формы и приемы обучения:

– предварительное самостоятельное знакомство студентов с изучаемым материалом;

– коллективный разбор большого количества практических заданий с использованием компьютера и мультимедиа-проектора;

– систематический блиц-опрос (тесты, «диктанты») в начале каждой лекции;

– использование мультимедийных презентаций, поясняющих в наглядной форме основные понятия объектно-ориентированного программирования;

– предоставление студентам после лекции всех ее материалов в электронном виде (основное содержание лекции, презентации, решенные задачи и составленные программы, дополнительные справочные материалы, электронные учебные пособия).

В ходе лабораторных занятий эффективными являются следующие методические приемы:

– выполнение учебных заданий разного типа:
  - на составление программы;
  - на поиск и исправление синтаксических и семантических ошибок в программе;
  - на определение результатов работы программы (выполнение трассировки);
  - на формулировку исходной задачи, которую решает представленная задача;

- на оптимизацию алгоритма и программы;

– составление учебных заданий самими студентами;

– применение не только индивидуальных, но и групповых форм организации учебной деятельности (например, коллективная разработка приложения, когда каждый студент реализует часть алгоритма в виде отдельного модуля (подпрограммы, класса); затем каждый из студентов пишет головную программу: осуществляет «сборку» приложения из разработанных модулей);

– «коллективно-конвейерный» способ решения задач: один студент начинает решение, затем преподаватель прерывает его и приглашает другого продолжить решение, и так до тех пор, пока задача не будет полностью решена (способ удобен тем, что повышает концентрацию внимания и обеспечивает эффективную работу со студентами, имеющими затруднения в освоении программирования).

В процессе подготовки лабораторного занятия преподаватель должен построить дидактически полную систему учебных заданий, обеспечивающих формирование необходимых знаний и умений и, если потребуется, их индивидуальную коррекцию. Заданий должно быть достаточно для того, чтобы, в случае необходимости, предложить студентам разные их наборы. Можно выделить четыре вида учебных заданий по объектно-ориентированному программированию [5, 6, 11, 12]:

– индивидуальные задания, выполнение которых позволит закрепить или проверить знание синтаксиса языка программирования (разработка одного-двух классов, реализация метода, алгоритма или его фрагмента и т.п.) – время выполнения 5-7 минут;

– фронтальные лабораторные работы, позволяющие на практике освоить объектно-ориентированное программирование – время выполнения 1,5-2 часа;

– индивидуальные и групповые проекты, выполнение которых дает студентам возможность освоить на практике объектную декомпозицию, что практически невозможно сделать в ходе лабораторных работ – длительность исполнения проекта составляет от одного месяца до целого семестра, примером тако-

го проекта может служить курсовая работа по дисциплине «Программирование»;

– долгосрочные коллективные проекты, выполнение которых организуется не отдельным преподавателем, а несколькими кафедрами университета (по прикладной тематике возможно участие кафедр педагогики, психологии и др.), основывается на тесном сотрудничестве участников; в ходе выполнения такого проекта студенты получают знания и опыт коллективной разработки программного обеспечения, учатся создавать повторно используемый программный код – длительность работы над таким проектом может составлять от года до двух-трех лет.

При этом, важным элементом обучения студентов программированию остается изучение алгоритмизации, основных типовых алгоритмов начиная от простейших и заканчивая, возможно, разбором алгоритмов, использующихся в выполнении заданий олимпиадного уровня (например, динамическое программирование, алгоритмы на графах и т.п.). Это позволяет студентам выполнять задания с использованием любого языка программирования на качественно новом уровне, в том числе успешно решать «олимпиадные» задачи.

При проведении экспериментальной работы по обучению студентов объектно-ориентированному программированию с одновременным использованием нескольких языков программирования были выявлены следующие проблемные области [5, 6]:

– методология объектно-ориентированного программирования представляется преподавателем (и в учебной литературе, особенно отечественной) весьма поверхностно, и, как следствие, у студентов не формируется глубокого понимания принципов и методологии объектно-ориентированного подхода;

– вследствие того, что на занятиях превалируют небольшие по объему учебные задания, студенты воспринимают объектную и алгоритмическую декомпозиции как абсолютно противоположные и взаимоисключающие, формируется представление, что в процессе анализа предметной области задачи нужно выбрать какую-то одну из них;

– недостаточная разработанность содержания учебных заданий и проектов, сложности с подбором большого числа однотипных заданий, а также заданий межпредметного характера, направленных на моделирование различных систем и процессов, в том числе на улучшение сопровождения программного обеспечения и создание повторно используемого программного кода.

Среди используемых методов и технологий обучения программированию будущего учителя информатики особое место занимает использование дистанционных технологий, поскольку их применение позволяет повысить интенсивность и результативность освоения программирования студентами любой формы обучения, как очной, так и заочной. Дистанционное обучение относится к одной из форм обучения, «это синтетическая, интегральная гуманистическая форма обучения, базирующаяся на использовании широкого спектра традиционных и новых информационных технологий и их технических средств, которые применяются для доставки учебного материала, его самостоятельного изучения, диалогового обмена между преподавателем и обучающимся, причем процесс обучения в общем случае некритичен к их расположению в пространстве и во времени, а также к конкретному образовательному учреждению» [3]. Целью использования дистанционных образовательных технологий является предоставление студенту возможности освоения образовательной программы в целом и, в частности, содержания конкретной учебной дисциплины, в удобное для него время и в удобном для него месте и темпе.

Были выявлены специфические особенности, которые имеет обучение студентов программированию средствами дистанционных технологий:

– поскольку определенную часть учебных заданий студент будет выполнять удаленно, целесообразно использовать непроприетарное программное обеспечение, в первую очередь, компиляторы и интегрированные среды разработки;

– помимо традиционных презентаций (в произвольном формате) в ходе учебных занятий требуются заготовки программного кода, которые будут ана-

лизироваться, обсуждаться, дополняться, изменяться преподавателем и студентами в ходе занятия и самостоятельной работы;

– целесообразно подготовить образцы выполнения типовых учебных заданий разных видов;

– так как результатом выполнения учебного практического задания, как правило, является программа (исходный программный код), для проверки его правильности рекомендуется применять проверяющую систему, которая в автоматическом режиме использует систему тестов и принимает (или не принимает) разработанную студентом программу.

В процессе обучения студентов программированию с применением дистанционных технологий реализуются различные модели учебного информационного взаимодействия [10], к которым в первую очередь можно отнести интеграцию аудиторных форм и использование коммуникационных сервисов и ресурсов информационной среды дистанционного обучения, поддержку учебного процесса для удаленных студентов, поддержку учебной и научно-познавательной проектной деятельности.

Для реализации дистанционных образовательных технологий в процессе обучения программированию в Институте математики, информатики и информационных технологий Уральского государственного педагогического университета применяются следующие программные средства:

– программное обеспечение для веб-конференций, обеспечивающее общение и сотрудничество через онлайн-доступ преподавателей, организаторов учебного процесса и студентов (Adobe Acrobat Connect Pro 8);

– средство организации совещаний и семинаров по сети в реальном времени, которое позволяет проводить презентации, обмениваться файлами, потоковым аудио, видео, а также организовать многопользовательские видеоконференции (Adobe Connect Pro Meeting – вебинар);

– электронный учебный портал (e-portal), использование которого позволяет организовать опосредованный обмен файлами для выдачи заданий сту-

дентам и отправку ими выполненных заданий обратно преподавателю (используется портал, созданный на базе платформы Sakai);

– проверяющая система для тестирования программного кода в режиме онлайн.

Система видеоконференцсвязи (ВКС) Adobe Connect Pro активно применяется в процессе подготовки студентов филиалов университета. При этом обеспечивается обратная связь с удаленными обучаемыми в режиме реального времени, а методы обучения и управления учебным процессом близки к традиционным аудиторным, хоть и имеют некоторую специфику.

На подготовительном этапе преподаватель подбирает учебно-методическое обеспечение, организует «собрание» (это термин ВКС), разрабатывает его макет, подготавливает опросы и оповещает студентов о собрании. Во время проведения интерактивного занятия преподаватель открывает собрание и управляет ходом занятия в соответствии с его планом, проводит запланированные опросы, обеспечивает обратную связь.

Несмотря на то, что система ВКС предоставляет возможность рассылки файлов (дополнительных материалов, заданий и пр.), опыт доказал, что для этой цели удобнее использовать учебный портал. Каждый из студентов университета, в том числе и удаленных, имеет логин и пароль, и может быть подписан преподавателем на нужный сайт учебной дисциплины, где размещены учебные задания и другие учебно-методических материалы.

Преимуществом такой организации учебного процесса является возможность предварительного знакомства студентов с учебными материалами, что делает последующее интерактивное занятие с помощью системы ВКС более эффективным – лекция превращается в консультацию и обсуждение наиболее сложных и значимых элементов содержания. Выполненные задания студенты имеют возможность отправить преподавателю на проверку как средствами учебного портала, так и через тестирующую систему. Второй вариант является предпочтительным, он избавляет преподавателя от необходимости читать и тестировать исходные коды студенческих программ.

В процессе обучения студентов программированию с применением средств дистанционных технологий были выявлены следующие организационные и методические проблемы:

– необходимость технологической подготовки преподавателя, особенно в области освоения возможностей системы ВКС;

– необходимость подготовки большого объема учебно-методических и других сопровождающих материалов, которые имеют специфику для использования их в процессе дистанционного обучения программированию;

– при работе с группой удаленных студентов в аудитории должен присутствовать специалист, способный оперативно решать возникающие технические и организационные проблемы;

– недостаточная разработанность содержания учебных заданий и проектов, сложности с подбором большого числа однотипных заданий, а также заданий, учитывающих специфику подготовки именно будущих инженеров, в том числе таких заданий, которые удаленные студенты могли бы выполнять коллективно;

– разработка учебно-методических материалов для использования в процессе обучения средствами дистанционных технологий имеет определенную специфику (дозированность учебного материала, его наглядность, интерактивность).

Опыт показал, что студенты не испытывают затруднений технологического характера при обучении с применением дистанционных образовательных технологий, в том числе на учебном занятии, проводимом с помощью системы ВКС.

Необходимо отметить, что освоение программирования будущим учителем информатики помимо узкопредметного и прикладного значения имеет и ярко выраженное развивающее и мировоззренческое значение. О влиянии изучения алгоритмизации и программирования на развитие мышления говорили еще классики информатизации образования А.П.Ершов, Ю.А.Первин,

Г.А.Звенигородский, С.Пейперт и др., утверждая, что при изучении программирования формируется операциональное, алгоритмическое, логическое, абстрактное, комбинаторное и другие виды мышления. Зарубежные и отечественные психологи (Ж.Пиаже, Я.А.Пономарев, Д.А.Поспелов и др.) подтверждали это в своих исследованиях. При освоении современных языков и технологий программирования у студентов происходит формирование и развитие некоторых специфических стилей мышления, а именно операционального, алгоритмического, объектного [7, 12, 21].

Рассмотрим более подробно влияние обучения программированию на формирование научного мировоззрения будущего учителя информатики.

Словари и энциклопедии трактуют мировоззрение как систему взглядов на объективный мир и место человека в нём, на отношение человека к окружающей его действительности и самому себе, а также как обусловленную этими взглядами основную жизненную позицию человека, его убеждения, идеалы, принципы познания и деятельности, ценностные ориентации. Мировоззрение – это предельное обобщение взглядов и представлений человека об окружающем мире, общее понимание мира, человека, общества, определяющее социально-политическую, философскую, религиозную, нравственную, эстетическую, научно-теоретическую ориентацию человека, его идеалы, убеждения, принципы познания, ценностные ориентации [18].

Под научным мировоззрением понимают мировоззрение, ориентирующееся в своих построениях на конкретные науки как на одно из своих оснований, особенно на их содержание – как материал для обобщения и интерпретации в рамках философской онтологии (всеобщей теории бытия) [17]. При этом сама наука как таковая мировоззрением, в строгом смысле этого слова, не является, так как, во-первых, она изучает саму объективную действительность, а не отношение человека к ней (а именно эта проблема является главным вопросом всякого мировоззрения), а во-вторых, любое мировоззрение является ценностным видом сознания, тогда как наука реализацией когнитивной сферы сознания, целью которой является получение знания о свойствах и отношениях раз-

личных объектов самих по себе. По мнению философов, особенно большое значение для научного мировоззрения имеет его опора на знание, получаемое в исторических, социальных и поведенческих науках, так как именно в них аккумулируется знание о реальных формах и механизмах отношения человека к действительности во всех ее сферах [21]. Однако несомненен вклад в формирование научного мировоззрения и естественных, и точных наук, в том числе и информатики.

Влияние изучения информатики (всех ее составляющих) на формирование научного мировоззрения обучаемого объясняется в первую очередь тем, что, по мнению современных ученых, вещество (материя), энергия и информация – это три важнейшие сущности нашего мира: из вещества мир состоит, энергией движется, информацией управляется. Эти фундаментальные понятия лежат в основе современной научной картины мира как целостной системы представлений об общих свойствах и закономерностях действительности [21].

Программирование, являясь одним из разделов информатики, также реализует мировоззренческую функцию.

В истории развития языков и технологий программирования выделяют несколько этапов:

- машинно-ориентированные языки низкого уровня;
- языки высокого уровня;
- языки структурного программирования;
- объектно-ориентированные языки.

Появление объектно-ориентированных языков является следствием эволюционного развития языков программирования и отражает не только общие тенденции в развитии информационных технологий, но и общие подходы к познанию окружающего мира. Возникновение и развитие объектно-ориентированного подхода к созданию и использованию средств информационных технологий объясняют следующими событиями, причем не только в сфере программирования и информационных технологий:

- прогресс в области развития архитектуры ЭВМ;

- развитие языков программирования;
- развитие методологии программирования, включая принципы модульности и скрытия данных;
- развитие теории баз данных;
- исследования в области искусственного интеллекта;
- достижения философии и теории познания.

При изучении программирования в вузе у студентов должно сформироваться представление о возможности двойственного взгляда на окружающую действительность – с точки зрения процессов (структурное программирование) и с точки зрения объектов (объектно-ориентированное программирование). Еще древние греки высказывали идею о том, что мир можно рассматривать в терминах как объектов, так и событий, выделив таким образом существование и алгоритмической, и объектной декомпозиции. В XVII веке Декарт отмечал, что люди обычно имеют объектно-ориентированный взгляд на мир. В XX веке эту тему развивала Рэнд в своей философии объективистской эпистемологии [20]. Марвин Мински предложил модель человеческого мышления, в которой разум человека рассматривается как общность различно мыслящих агентов – объектов [19]. Он доказывает, что только совместное действие таких агентов приводит к осмысленному поведению человека.

Алгоритмическая декомпозиция понимается как разделение алгоритмов, причем каждый модуль выполняет один из этапов общего процесса (профессор Э. Дейкстра [9]). Сущность объектно-ориентированной декомпозиции состоит в разделении системы на элементы (объекты), где критерием разделения является принадлежность элементов к различным абстракциям (типам) данной предметной области. В объектно-ориентированной декомпозиции мир задачи представляется совокупностью автономных действующих лиц (объектов), которые взаимодействуют друг с другом, чтобы обеспечить поведение системы, соответствующее более высокому уровню. Каждый объект модели обладает своим собственным поведением и моделирует некоторый объект реального мира, т.е. является вполне осязаемой «вещью», которая демонстрирует вполне оп-

ределенное поведение. В отличие от алгоритмической декомпозиции, в объектно-ориентированной модели не присутствуют независимые алгоритмы, все действия существуют только как операции над определенными объектами, точнее, над их полями. В объектно-ориентированном программировании функциональный поток заменяется передачей сообщений между объектами, которые вызывают изменения состояния. Таким образом, объектно-ориентированное программирование – это крайне естественный подход, поскольку структура программ непосредственно отражает структуру задачи. Более того, в моделируемых задачах обычно понятно, что является объектами. В частности, это могут быть машины на улице, механизмы производственной линии, геометрические фигуры и т.д.

Мировоззренческое значение освоения объектно-ориентированного программирования заключается в реализации следующей связи между понятиями: объект – источник информации – изучение объекта через его свойства – изменение свойств объекта через его методы – поведение объекта при взаимодействии с внешней средой [21].

Возникает вопрос: какая декомпозиция сложной системы рациональнее – по алгоритмам или по объектам? Анализ обозначенной проблемы дает возможность утверждать, что важны оба аспекта. Но систему невозможно сконструировать сразу двумя ортогональными способами: следует начинать с объектной декомпозиции, так как она имеет следующие преимущества перед алгоритмической:

- позволяет повторно использовать общие механизмы, что приводит к существенной экономии выразительных средств, а также уменьшает размер программы;
- объектно-ориентированная система более гибка, проще изменяется и эволюционирует;
- объектная декомпозиция помогает разобраться в сложной системе, предлагая, как правило, более разумные варианты решения поставленной задачи.

После объектной декомпозиции, используя полученную структуру, следует рассмотреть проблему и с другой точки зрения – алгоритмической. Таким образом, объектно-ориентированная декомпозиция не отрицает декомпозицию алгоритмическую, а включает ее в себя, подчиняя построенной объектной модели.

Для наиболее полной реализации мировоззренческой функции курса программирования в вузе можно сформулировать следующие рекомендации по построению содержания курса и применению методов обучения и методических приемов.

Начинать обучение программированию следует не со структурных, а с объектно-ориентированных языков программирования (Object Pascal, Java, C++) [2,5,6]. На начальном этапе освоения объектно-ориентированного программирования классы, которые разрабатывают студенты, могут (и должны) быть простейшими: несколько полей (элементарных, а не структурированных типов), конструктор и несколько методов, реализующих линейные алгоритмы обработки данных (полей).

На начальном этапе важно сформировать у обучаемых представление о классе как совокупности полей и методов, научить их объявлять класс как модель некоторой предметной области задачи и создавать объекты классов. При этом важны как синтаксическая, так и семантическая сторона используемых конструкций языка программирования. Студент должен не просто запомнить правила синтаксиса, но осознать их смысловую обусловленность [1,5,12].

На этом этапе можно придерживаться следующих методических рекомендаций:

- при решении любой учебной задачи перед реализаций класса на конкретном языке программирования сначала спроектировать его, а именно:
    - выделить поля и определить их типы: у обучаемого должно сформироваться четкое представление о том, что поля – это данные об объекте, они являются независимыми (если некоторое данное зави-

сит от другого, то это уже не поле, а метод – функция), и вспомогательные (локальные) переменные не являются полями;

- продумать формальные параметры для конструктора (или конструкторов) класса;
- на основе поставленной задачи выделить методы класса, обязательно указав, какие из них будут являться модификаторами (т.е. будут изменять значения полей), а какие – селекторами (т.е. будут сообщать о состоянии полей); особое внимание при этом необходимо уделить формированию представления о методе как подпрограмме, инкапсулированной в класс, что будет вступать в некоторое противоречие с усвоенными ранее (например, при изучении структурного программирования в школьном курсе информатики) представлениями, в частности, о формальных параметрах методов;

- прослеживать жизненный цикл объекта от момента его создания с помощью конструктора до удаления из памяти с помощью деструктора (если он имеется);
- предлагать студентам обратные задачи: на основе представленного описания класса предложить возможные варианты предметной области, которая может быть представлена с помощью такого описания.

Следующим этапом обучения является освоение основных алгоритмических конструкций и их реализации в конкретном языке программирования, а также системы его типов данных. По сути, при этом рассматривается структурное программирование, без него невозможно реализовать классы для решения более сложных задач. При этом рекомендуется:

- применять структурированные типы данных (в первую очередь, массивы) как для организации полей класса, так и для организации объектов класса;

- аналогичным образом, использовать базовые алгоритмические конструкции как при описании методов, так и для манипулирования объектами класса;
- обсуждать со студентами возможность двойственного взгляда на окружающую действительность: с точки зрения процессов и с точки зрения взаимодействующих объектов. Самое простое задание: представьте учебное занятие как процесс, т.е. алгоритм, имеющий начало, реализацию базовых алгоритмических конструкций (линейной, ветвления, цикла) и конец, и как взаимодействие объектов, принадлежащих различным классам: люди (студенты и преподаватель), компьютерная техника, средства обучения и т.п.

Затем целесообразно рассмотреть различные отношения между классами, в первую очередь, наследование, а затем и композицию классов. При этом решение любой учебной задачи необходимо начинать с построения прежде всего объектной модели ее предметной области, и только потом алгоритмической.

Представляется целесообразным анализировать на занятиях историю развития языков программирования, раскрывать студентам сущность понятия «парадигма программирования».

Рекомендуем рассматривать технологию объектно-ориентированного программирования на методологическом уровне (основные идеи и принципы), а не только на прикладном. Содержание понятий «объектно-ориентированный метод» и «объектно-ориентированный подход» существенно шире, чем просто «объектно-ориентированное программирование». Они включают философию разработки систем программирования, извлечение знаний, анализ требований, моделирование предметной области, проектирование систем, проектирование баз данных и многие другие вопросы. Конечно, в начальном курсе программирования не найдет воплощения указанная философия и ее использование для решения проблем, возникающих при создании информационных систем. Однако рассмотрение этих вопросов и иллюстрация их конкретными примерами позволит в дальнейшем будущему ИТ-специалисту применить эти теоретические знания в профессиональной деятельности, быть в курсе актуального состояния

языков и технологий программирования. Именно поэтому целесообразно анализировать различия в содержании понятий «объектно-ориентированное» (object-oriented), «компонентно-ориентированное» (component-oriented), «основанное на объектах» (object-based) и «основанное на классах» (class-based) программирование, а также в их разработке и анализе.

Рассматриваемые методологические основы объектно-ориентированного подхода можно закрепить при выполнении учебных заданий [4, 13], подобных приведенным ниже.

Задание 1. Какая концепция искусственного интеллекта является наиболее близкой к понятию объекта?

а) слот

б) механизм вывода

в) база знаний

г) фрейм

д) фацет

е) правило

Задание 2. Чем различаются перечисленные ниже понятия (сформулируйте по одному предложению для каждого пункта)?

а) экземпляр и класс

б) тип данных и класс

в) класс и роль

г) тип объекта и тип сущности

д) класс и компонент

е) динамическое связывание и полиморфизм

ж) обобщение и наследование

з) наследование и классификация

Задание 3. Прокомментируйте следующее высказывание: «Объектная технология умерла. Ей на смену пришла компонентно-ориентированная разработка».

Выполнение подобных заданий будет способствовать формированию у студентов обобщенных взглядов и представлений об окружающем мире [4, 15], его общему пониманию, а, значит, и формированию научного мировоззрения.

В качестве еще одной рекомендации сформулируем целесообразность применения в процессе обучения как визуальных, так и невизуальных сред программирования [2,5,6]. Это позволит сформировать у студентов представление о независимости результатов решения задачи от используемого инструментария, умение использовать различные инструменты для решения сходных задач, а также умение осваивать новое программное обеспечение.

Представляет интерес анализ курсовых работ, выполняемых студентами второго курса Института математики, информатики и информационных технологий Уральского государственного педагогического университета. Курсовые работы выполняются по дисциплинам «Технологии программирования», «Языки и технологии программирования» (в зависимости от направления подготовки студентов). Около 30% студентов при выполнении курсовой работы самостоятельно осваивают новый для них инструментарий: языки, среды программирования, библиотеки, интерфейсы и т.п. Подавляющее большинство студентов (более 80%) грамотно выполняют объектно-ориентированный анализ предметной области и объектно-ориентированное проектирование, представляют методологически правильно разработанный и устойчиво функционирующий программный продукт. Конечно, эти результаты в первую очередь свидетельствуют о качестве предметной подготовки будущих учителей информатики в области программирования. Но также они говорят о том, что студенты овладели

некоторыми способами и инструментами моделирования окружающего мира, что оказало влияние и на их научное мировоззрение.

Современное миропонимание – важный компонент человеческой культуры. Очевидно, что каждый культурный человек должен представлять, как устроен мир, в котором он живет. Изучение объектно-ориентированного программирования дает возможность рассмотреть окружающий мир с двух разных точек зрения: как совокупность объектов и как совокупность процессов. Структурное и объектно-ориентированное программирование являются двумя инструментами моделирования окружающего мира вообще и предметной области решаемой задачи в частности. Следовательно, изучение программирования активно участвует в формировании мировоззрения человека – совокупности его обобщенных взглядов на мир в целом и на свое отношение к этому миру.

Наблюдение за деятельностью студентов при проведении эксперимента, анализ выполненных индивидуальных заданий, курсовых работ, результатов участия в олимпиадах и конкурсах различных уровней привели к выводу о возможности и целесообразности организации обучения программированию будущего учителя информатики на основе объектно-ориентированного подхода с использованием нескольких языков программирования. При этом формируется предметная составляющая профессиональной подготовки будущего учителя, формируется и развивается абстрактное, логическое и алгоритмическое мышление, опосредованно оказывается влияние и на его методическую компетентность, так как содержание и методы обучения в вузе переносятся затем начинающим учителем в школу, в практику его профессиональной деятельности.

ЛИТЕРАТУРА

1. Алексеевский П. И. Применение средств управления версиями для коллективной работы студентов над проектом компьютерной игры. // Педагогическое образование в России. 2012. № 6. С. 51-54.

2. Алексеевский П. И., Лапенок М. В. Выбор программного обеспечения для проведения практических занятий по программированию на С и С++ // Информатика и образование. 2010. № 2. С. 117-119.

3. Андреев А.А. К вопросу об определении понятия «дистанционное обучение». – URL: http://www.e-joe.ru/sod/97/4_97/st096.html.

4. Газейкина А. И. Обучение будущего учителя информатики конструированию учебных заданий, направленных на формирование метапредметных результатов обучения // Педагогическое образование в России. 2014. № 8. С. 159-164.

5. Газейкина А. И. Обучение программированию будущего учителя информатики // Педагогическое образование в России. 2012. № 5. С. 45-48.

6. Газейкина А. И. Обучение программированию будущих ИТ-специалистов с применением дистанционных технологий / Подготовка молодежи к инновационной деятельности в процессе обучения физике, математике, информатике: материалы международной научно-практической конференции. Урал. гос.пед.ун-т. Екатеринбург, 2014. С. 33-37.

7. Газейкина А. И. Стили мышления и обучение программированию студентов педагогического вуза. URL: http://ito.edu.ru/2006/Moscow/I/1/I-1-6371.html (дата обращения 30.06.2015).

8. Газейкина А. И. Формирование научного мировоззрения будущего ИТ-специалиста в процессе обучения программированию // Педагогическое образование в России, 2015. № 7 – Екатеринбург, УрГПУ – С. 36-41.

9. Дейкстра Э. Дисциплина программирования. М.: Мир, 1978. – 275 с.

10. Лапенок М.В. Теоретические модели осуществления учебного информационного взаимодействия в информационной среде дистанционного обучения // Педагогическое образование в России, 2012, № 2. 214-217.

11. Петров А.Н. Основные подходы к обучению студентов объектно-ориентированному программированию и проектированию // Фундаментальные исследования. – 2008. – № 4 – стр. 80-82 [Электронный ресурс] URL:

www.rae.ru/fs/?section=content&op=show_article&article_id=7780826 (дата обращения: 26.09.2012).

12. Рожина И. В. Обучение учащихся объектно-ориентированному программированию и технологии визуального проектирования в базовом курсе информатики. Дисс. канд. пед. наук. Екатеринбург, 2002. – 150 с.

13. Семенова И. Н., Кузьмина Т. А. Конвенциально-ролевая рефлексия как механизм проявления автологичности методов обучения в процессе педагогического образования студентов // Педагогическое образование в России. 2012. № 2. С. 151-154.

14. Стариченко Б.Е., Шеметова А.Д. Совершенствование информационно-технологической подготовки студентов на основе системно-объектного подхода // Образование и наука. Известия УрО РАО, 2009, № 4(61). С. 78-91.

15. Усольцев А. П., Курочкин А. И. Концепция развивающего обучения при построении системы задач как средство решения современных образовательных проблем // Педагогическое образование в России. 2013. № 6. С. 248-251.

16. Федеральный государственный стандарт высшего профессионального образования по направлению подготовки 050100 Педагогическое образование [Электронный ресурс]. http://минобрнауки.рф/документы/924

17. Философия науки: Словарь основных терминов. М.: Академический Проект. С.А. Лебедев. 2004. – 320 с.

18. Философия: Энциклопедический словарь. М.: Гардарики. Под редакцией А.А. Ивина. 2004. – 1072 с.

19. Minsky M. The Emotion Machine: Commonsense Thinking, Artificial Intelligence, and the Future of the Human Mind. – Simon & Schuster Paperbacks, 2007. – 400 p.

20. Rand A. Introducing Objectivism, in Peikoff, Leonard, ed. The Voice of Reason: Essays in Objectivist Thought. Meridian, New York, 1990. – 228 p.

21. Starichenko B. E. Conceptual basics of computer didactics: Monograph – Yelm, WA, USA: Science Book Publishing House, 2013. – 184 p.

3. СОВРЕМЕННОЕ СОСТОЯНИЕ ИССЛЕДОВАНИЙ В ОБЛАСТИ СОЗДАНИЯ И ИСПОЛЬЗОВАНИЯ ЭЛЕКТРОННЫХ ОБРАЗОВАТЕЛЬНЫХ РЕСУРСОВ

В соответствии с Концепцией долгосрочного социально-экономического развития Российской Федерации стратегической целью в области образования является повышение доступности качественного образования, соответствующего современным потребностям общества и каждого гражданина. Возрастают требования к подготовке критически мыслящей личности, способной к непрерывному обновлению своих знаний, быстрому переучиванию и смене области применения своих способностей. А для этого необходимо создание новых условий и методик обучения, которые и являются основой новых образовательных технологий [1]. К таким инновационным технологиям относятся дистанционные образовательные технологии. Импульсом для их развития послужило принятие в 1995 году «Концепции создания и развития системы дистанционного образования в России» и внедрение в 1999 году Государственной программы информатизации образования. «Высшей целью создания и развития системы дистанционного образования является предоставление школьникам, студентам, гражданским и военным специалистам, безработным, самым широким кругам населения в любых районах страны и за ее рубежами равных образовательных возможностей» [23]. В документах Института информатизации ЮНЕСКО среди наиболее важных направлений формирования перспективной системы образования обозначены:

– повышение качества образования путем фундаментализации и применения различных подходов с использованием информационных технологий;

– реализация концепции опережающего образования, ориентированного на новые условия уже формирующегося в передовых странах мира информационного общества;

– широкое внедрение методов инновационного и развивающегося образования, ориентированного на раскрытие творческого потенциала личности;

– повышение доступности качественного образования для широких слоев населения путем развития систем дистанционного обучения на основе современных информационных и телекоммуникационных технологий [11].

С 2001 г. Правительством Российской Федерации было принято несколько программ («Электронная Россия», «Развитие единой образовательной информационной среды», и ряд других), которые должны были изменить ситуацию с оснащением учебных заведений компьютерами и подключением к сети Интернет. В [6] отмечается, что в результате реализации этих программ российские учебные заведения почти сравнялись по основным показателям оснащенности средствами информационных и коммуникационных технологий с европейскими образовательными учреждениями, о чем свидетельствуют данные различных международных социологических исследований. Одновременно с этим была определена новая стратегия на модернизацию образовательной системы в целом, основными задачами которой являются возрождение и развитие лучших традиций российского просвещения, упрочение позиций России в ряду высокообразованных стран мира и ее интеграция в мировое образовательное сообщество. В «Концепции модернизации российского образования на период до 2010 года», принятой Правительством РФ в декабре 2001г., говорится о роли средств информационных и коммуникационных технологий [3].

За период 2002 - 2010 года были разработаны ЭОР и учебно-методические материалы для обучения школьников начальной школы, среднего и старшего звена с привлечением многих издательств (ЗАО "Просвещение Медиа", "1С", ОАО "Издательство "Просвещение", ООО "Физикон", "Кирилл и Мефодий", "Дрофа" и др.). Разработаны учебно-методические материалы и сайты для подготовки преподавателей системы Интернет-обучения, на базе которых прошли обучение более 500 педагогов-кураторов и около 60 учителей-кураторов сетевого информационного взаимодействия для обучения

школьников с помощью ДОТ. Обучено 4 000 учеников в рамках эксперимента по разработке и апробации модели обучения школьников в реальных условиях на местах с использованием ДОТ.

Разработанные учебно-методические материалы охватывают программу всех школьных дисциплин.

В Каталоге ресурсов для образования [4] представлены:

– коллекции электронных образовательных ресурсов, созданные и размещенные в сети Интернет за последние годы в рамках различных образовательных программ, выполняемых на федеральном уровне, в том числе: Федеральный центр информационно-образовательных ресурсов (http://fcior.edu.ru, http://eor.edu.ru), Информационная система «Единое окно доступа к образовательным ресурсам» (http://window.edu.ru), Единая Коллекция цифровых образовательных ресурсов (http://school-collection.edu.ru);

– общероссийские образовательные порталы, в том числе: сайт Министерства образования и науки РФ (http://www.mon.gov.ru), сайт Рособразования (http://www.ed.gov.ru), Федеральный портал «Российское образование» (http://www.edu.ru), Российский общеобразовательный портал (http://www.school.edu.ru), портал информационной поддержки Единого государственного экзамена (http://ege.edu.ru);

– каталог учебных изданий, оборудования и электронных образовательных ресурсов для общего образования (http://ndce.edu.ru), школьный портал (http://www.portalschool.ru);

– федеральный портал «Информационно-коммуникационные технологии в образовании» (http://www.ict.edu.ru);

– региональные образовательные порталы и сайты органов управления образованием (более 55 республиканских, окружных, областных и городских образовательных сайтов).

В качестве основной тенденции развития ЭОР в мире [5] рассматривается разработка ЭОР модульной структуры, в которых по каждому учебному предмету разработаны так называемые открытые образовательные модульные муль-

тимедиа системы (ОММС), состоящие из тематических элементов. Для каждого тематического элемента разработаны три типа электронных учебных модулей: модуль получения информации; модуль практических занятий; модуль контроля (в общем случае – аттестации). При этом каждый учебный модуль автономен, представляет собой законченный интерактивный мультимедиа продукт, нацеленный на решение определенной учебной задачи. Для каждого учебного модуля разрабатываются аналоги – вариативы, посвященные одному и тому же тематическому элементу данной предметной области. Структура ЭОР с вариативами исполнения электронных учебных модулей в ОММС обеспечивает возможность выбора их оптимальной с персональной точки зрения комбинации для курса по предмету; т.е. обеспечивает возможность построения авторского учебного курса преподавателем и создания индивидуальной образовательной траектории учащегося. В [7] также рассматривается развитие ЭОР различных типов с целью решения педагогических задач по организации информации, формированию практических умений, контролю знаний. Однако анализ показывает, что в большинстве случае все ресурсы независимо от указанного типа носят комплексный характер и содержат информационные компоненты и компоненты для самоконтроля. Анализ ресурсов, размещенных в Федеральном центре информационных образовательных ресурсов, показывает, что все ЭОР, размещенные во ФЦИОР сделаны по единому шаблону, большинство ресурсов требует установки специального ОММС плеера, работающего в ограниченном числе операционных систем.

Анализ ресурсов, размещенных в Единой коллекции цифровых образовательных ресурсов, показывает следующее распределение ЭОР для общего образования по предметам: русскому языку - более 4280, литературе - более 6480, иностранным языкам - более 6340, математике - более 17100, истории - более 16400, обществознанию - более 900, географии - более 2230, естествознанию - более 1980, физике - более 8880, химии - более 2800, биологии - более 1330, изобразительному искусству - более 6200, музыке - более 6450, технологии - более 330. Общее число посетителей за время работы Единой коллекции, по

информации Национального фонда подготовки кадров, превысило 50 миллионов пользователей, среднее ежемесячное число пользователей превышает миллион человек, число обращений доходит до 10 миллионов [4]. Учитывая разнонаправленность ЭОР и различие технологий их представления, по мнению академика А.Н. Тихонова целесообразна разработка единого методического и методологического подхода использования ЭОР в учебной деятельности [8]. Число скачиваний каждого ресурса для разных предметов колеблется в приделах от 50 до 4000 скачиваний, что говорит о том, что даже наиболее популярные ресурсы попали менее чем в 10% школ России [4]. Для широкого внедрения электронных образовательных ресурсов в практику массовой общеобразовательной школы необходимо обеспечить возможность редактирования таких ресурсов с целью их применения в авторских методиках обучения.

Анализ мнения преподавателей-предметников общеобразовательных школ о качестве ЭОР показывает: несмотря на то, что ЭОР, помещаемые в ФЦИОР, проходят экспертную оценку, качество этих ресурсов далеко от совершенства [8].

Анализ зарубежных официальных сайтов образовательного назначения показывает, что сайты носят, в основном, коммерческий характер, и представлены набором платных образовательных услуг. Например, главной целью специализированного сайта Британского агентства по образованию, связям и технологиям является организация и осуществление совместной работы школ, спонсоров и коммерческих предприятий в использовании сетевых технологий для выполнения образовательных стандартов (http://www.becta.org.uk/). Подробный анализ разделов рассматриваемого сайта показывает целесообразность использования представленной информации лишь для системы образования Великобритании. Информация, носящая в основном коммерческий характер, не может иметь практического применения в системе образования других стран, в частности, России.

Анализ опыта обучения учащихся общеобразовательных школ использованием ЭОР и ДОТ, представленный в изданиях ГНИИ ИТТ «Информика» под

редакцией А.Н.Тихонова [9], показал, что в целом по России более 90% общеобразовательных школ используют современные технологии подключения к сети интернет (ассиметричный цифровой канал передачи данных, волоконно-оптические линии связи, спутниковый и радиодоступ), которые позволяют обеспечить полноценную работу в сети Интернет нескольких учебных классов, в том числе одновременную загрузку веб-ресурсов, ЭОР, работу с ФЦИОР [9]. Однако для 70 % школ ограничение скорости со стороны провайдера (128 Кбит/с), не позволяет использовать ЭОР из Интернета несколькими учащимися одновременно. Это говорит о том, что необходимо ориентироваться не на работу с удаленными серверами, а на создание внутришкольной образовательной среды обучения, ресурсы которой были бы доступны школьникам с домашнего компьютера.

По данным Всероссийского центра изучения общественного мнения (ВЦИОМ) школьники большую часть времени (от 60 до 80%), проводимого за компьютером и в Интернете, тратят на общение и игры. При этом в школе учащиеся используют компьютер для решения учебных задач (71% – сельские и 68% – городские школьники). Можно отметить, что использование ресурсов сети Интернет в школе связано с активной деятельностью школьников по поиску учебной информации. При этом одним из основных направлений использования домашнего компьютера и Интернета в учебных целях является работа в системах дистанционного обучения [9]. Привлекательной для школьников является перспектива участия в глобальных детских сетевых проектах и интерактивное общение со своими сверстниками, получение информации из различных источников со всего мира. Это может быть основная или любая дополнительная информация при учебном проектировании, выполнении домашнего задания или изучении новой темы, а также, возможно, законодательные или нормативно-правовые документы.

Анализ более 2 тыс. анкет руководителей ОУ из разных регионов РФ [9] показал, что большинство школ РФ оборудованы компьютерными классами,

средствами мультимедиа и интерактивными досками, но не имеют возможности одновременной работы учащихся в сети Интернет.

В связи с этим, для повышения уровня предоставляемых образовательных услуг, реализованных на основе вычислительных сетей, целесообразна разработка и применение в общеобразовательной школе такой организации учебного процесса, которая позволит учащимся использовать для загрузки электронных образовательных ресурсов в основном домашний компьютер, подключенный к Интернету. Тогда как коллективную работу в режиме реального времени имеет смысл применять только на этапах промежуточной или итоговой аттестации. Для этих целей представляется актуальной задача совершенствования используемых технологий дистанционного обучения с использованием Интернета, создания и развития соответствующих информационных сред и методических материалов**.**

Федеральный закон об образовании не выделяет дистанционное обучение как самостоятельную форму обучения, хотя и разрешает использование дистанционных образовательных технологий при реализации образовательных программ [10,ст.14].

Среди лиц, остро нуждающихся в образовательных услугах посредством ДОТ, можно выделить следующие группы школьников:

– учащиеся старших классов, не имеющие возможности получить образовательные услуги выбранного профиля в рамках традиционной классно-урочной формы;

– талантливые и продвинутые учащиеся, стремящиеся получить дополнительные, более глубокие и более широкие, знания;

– учащиеся, имеющие медицинские ограничения для получения регулярного образования в стационарных условиях (нуждающиеся в обучении на дому);

– учащиеся старших классов, готовящиеся к поступлению в вузы;

– учащиеся старших классов, желающие участвовать в проектной, олимпиадной, иной самостоятельной познавательной, творческо-поисковой деятельности;

– учащиеся, вынужденные пропускать по разным причинам занятия на короткий период времени, например, по болезни и другим причинам, вследствие климатических, социально-экономических условий, по другим причинам, связанным с жизнедеятельностью самих школьников или учителей;

– учащиеся из семей мигрантов и беженцов, имеющие разницу в учебных планах образовательных программ;

– учащиеся специализированных учреждений общего образования, типа городских центров образования, вечерних школ и т.п.

Однако, реализация дистанционного обучения невозможна без наличия подготовленных к работе с технологиями дистанционного обучения педагогических кадров. Для решения задач образования в условиях его информатизации необходимо сформировать у учителя – предметника готовность к реализации дистанционного обучения предмету (естественнонаучному, гуманитарному или др.), которая в настоящее время является одним из элементов целостной готовности педагога к профессиональной деятельности в условиях информатизации общества и образования.

Одним из аспектов подготовки учителей к профессиональной деятельности в условиях информатизации образования является целенаправленное формирование у них способности использовать в учебном процессе ИКТ, адекватные целям, задачам, формам и условиям организации учебного процесса в конкретном учебном заведении. А в случае, когда педагогическая деятельность осуществляется в условиях функционирования ИСДО, в процессе подготовки учителя необходимо сформировать способность выполнения профессиональной деятельности с использованием сервисов системы дистанционного обучения.

Для этих целей в образовательных программах подготовки педагогических кадров должно быть уделено внимание вопросам использования дистан-

ционных образовательных технологий в школе, создания электронных образовательных ресурсов в условиях функционирования ИСДО.

Основными содержательными линиями подготовки учителя должны стать: общие вопросы информатизации образования; технологии создания ЭОР ИСДО и их использования в учебном процессе; оценка педагогико-эргономического качества ЭОР; теоретические аспекты организации учебного процесса в ИСДО; возможности систем дистанционного обучения при организации учебного процесса. Необходимо определить организационные формы подготовки учителей, включающие: теоретические, практические и лабораторные занятия, на которых формируются знания и практические умения учителей по применению сервисов системы дистанционного обучения для создания ЭОР, для реализации вариативных форм и методов обучения в ИСДО.

На наш взгляд учебная дисциплина должна включать следующие разделы.

2. Теоретические основы информатизации образования.
   - 2.1. Понятийный аппарат дистанционного обучения и информатизации образования в целом: информационно-коммуникационные технологии, дистанционные образовательные технологии, информационная среда дистанционного обучения.
   - 2.2. Дидактические возможности информационных и коммуникационных технологий: интерактивный диалог, компьютерная визуализация учебной информации, компьютерное моделирование, архивирование, автоматизация процессов вычислительной, информационно-поисковой деятельности, автоматизация процессов информационно-методического обеспечения, организационного управления учебной деятельностью.
   - 2.3. Педагогически значимые цели их реализации: развитие личности обучающегося, реализация социального заказа, интенсификация учебного процесса.
   - 2.4. Педагогические основания развития информатизации образования: изменение структуры и содержания информационного взаимодействия,

изменение структуры представления учебного материала, состава и содержания учебно-методического обеспечения образовательного процесса, развитие информационно-коммуникационной предметной среды как условий взаимодействия между участниками образовательного процесса.

3. Электронные образовательные ресурсы: характеристики, технологии создания и применения для дистанционного обучения школьников.
   3.1. Виды электронных образовательных ресурсов, в том числе ЭОР ИСДО.
   3.2. Категории пользователей и тематика их интересов.
   3.3. Программные средства разработки ЭОР.
   3.4. Содержательно-методические, технико-технологические, дизайн-эргономические требования к качеству ЭОР; требования к уровню реализации технологии мультимедиа с учетом условий интерактивного взаимодействия пользователей с ЭОР.
   3.5. Методы оценки педагогико-эргономического качества ЭОР.
   3.6. География размещения образовательных ресурсов, статистика распределения ресурсов по учебным дисциплинам.
   3.7. Информационная среда дистанционного обучения общеобразовательной школы. Структура и технологии создания образовательного портала.
4. Организация учебного процесса в условиях функционирования информационной среды дистанционного обучения.
   4.1. Систематизация видов организации учебного процесса с использованием ЭОР и ДОТ: консультирование; дифференцированное обучение; обучение в распределенных группах; самообучение; обучение в компьютерных классах с расширенным набором периферийных устройств; мобильное обучение.
   4.2. Интегративная организация учебного процесса: организация учебного процесса в классно-урочной форме с использованием ЭОР ИСДО; организация учебного процесса в период пропусков занятий учащимися;

организация учебной и исследовательской, проектной, олимпиадной деятельности школьников во внеурочной время.

4.3. Нормативно-правовое обеспечение использования дистанционных образовательных технологий в учебном процессе школы.

4.4. Структура учебно-методического комплекса по предмету.

4.5. Категории и базовый функционал участников образовательного процесса при использовании технологий дистанционного обучения в ИСДО: учащегося; учителя-куратора сетевого информационного взаимодействия; создателя курсов; администратора ИСДО; других авторизованных пользователей (родителей, руководство школы).

4.6. Обзор программного обеспечения для создания учебных материалов и ведения дистанционного обучения: система дистанционного обучения Moodle; система дистанционного обучения Naulearning; система Docent; система BlackBoard; система дистанционного обучения LearningSpace 5.0; система дистанционного обучения WebCT; система дистанционного обучения eLearning Server 3000; система дистанционного обучения ОРОКС.

3.7. Педагогические технологии дистанционного обучения: обучение в сотрудничестве (collaborative learning); технологии кооперативного обучения (cooperative learning); телекоммуникационные проекты; технологии проблемного обучения; индивидуальное и дифференцированное обучение; модульное обучение; игровые технологии (ролевые и деловые игры, ситуационный анализ); метод "мозгового штурма"; менторство (индивидуальное наставничество); парное обучение.

5. Основы работы в информационной среде дистанционного обучения.

5.1. Структура и компонентный состав системы дистанционного обучения (на примере Sakai, Naulearning).

5.2. Портал дистанционного обучения: создание портала; создание разделов портала; создание документа в разделе портала; создание новостей портала; копирование, перемещение и удаление объектов портала.

5.3. Учебный курс: создание учебного курса; системные папки курса; управление инструментами курса (новости, форум, события); создание главы курса; создание урока; работа с кадрами; прикрепление графических файлов и файлов flash-анимации к кадру.

5.4. Создание элементов ЭОР: текстовых материалов; упражнений и заданий; экзаменационных тестов; вопросов для самопроверки; мультимедийных иллюстративных материалов.

5.5. Организация и управление процессом обучения: регистрация пользователей в системе; подача и обработка заявок пользователей на курс; работа с форумами; просмотр запланированных событий; сервисы «голосование», чат; контроль успеваемости в подсистеме обучения; отчетность и статистика по проектам обучения.

Такой курс (объемом около 72 часов) может быть внедрен в рамках вариативной части учебного плана для подготовки учителя в любой предметной области.

ЛИТЕРАТУРА

1. Ибрагимов И.М. Информационные технологии и средства дистанционного обучения. 2-е изд. М.: Издательский центр «Академия», 2007. – 336 с.
2. Концепции модернизации российского образования на период до 2010 года // Президент России – URL: http://archive.kremlin.ru/text/docs/2002/04/57884.shtml (дата обращения: 9.03.2010)
3. Концепция создания и развития единой системы дистанционного образования в России // Теории и концепции ДО – URL: http://de.unicor.ru/science/groundwork/concept.html (дата обращения: 20.02.2012)
4. Описание потребностей основного общего образования в электронных образовательных ресурсах (Проект) // Электронные образовательные

ресурсы – URL: http://www.eor-np.ru/node/2699 (Дата обращения 28.01.2012)

5. Осин А.В. Открытые образовательные модульные мультимедиа системы – М.: Агентство ”Издательский сервис”, 2010. – 328 с.
6. Реморенко И.М. Департамент государственной политики и нормативно-правового регулирования в сфере образования // Методические рекомендации по проведению августовских педагогических совещаний работников образования "Актуальные задачи современной модели образования" N 03-946 от 8 мая 2008 г.
7. Смольникова И. А. Структуризация основных требований к ЭОР // электронный журнал "Вопросы Интернет образования" № 97. – URL: http://vio.uchim.info/Vio_97/default.htm (дата обращения: 10.05.2013)
8. Тихонов А.Н. Образовательные ресурсы сети интернет для основного общего и среднего (полного) общего образования. М., – 80 с. // Образовательные ресурсы сети интернет – URL: http://catalog.iot.ru/pdf/ Catalog_vol5.pdf (Дата обращения: 10.02.2013)
9. Тихонов А.Н. Оценка уровня информатизации общеобразовательных учреждений России (информационно-аналитические материалы). М.: Гос. НИИ информационных технологий и телекоммуникаций «Информика». 2009, 64 с.
10. Федеральный закон «Об образовании в Российской Федерации» от 26 декабря 2012 г. №273-ФЗ // КонсультантПлюс. URL: http://www.consultant.ru/law/hotdocs/23125.html#.UacO4tLiGrk (Дата обращения: 04.02.2013)
11. Tinker R., Galvis A., Zucker A. 1:1 Computing in support of science and mathematics education. Recommendations for Large-Scale Implementations, Concord, MA, 2007.

4. ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ НА ПРИМЕРЕ СОЗДАНИЯ ПРОГРАММЫ «ШАХМАТЫ»

В этой статье приведены общие подходы к созданию приложения “Шахматы”, в котором продемонстрировано применение современной методики объектно-ориентированного программирования. В качестве среды разработки выбрана среда визуальноного программирования Delphi. Это связано с тем, что данный продукт широко распространен при обучении программированию в учебных заведениях, а сама игра является замечательным примером, к которому может быть применен именно объекно-ориентированный подход.

Объектно-ориентированное программирование (ООП) - современная технология, позволяющая разрабатывать сложные системы, устойчивые по отношению к ошибкам и допускающие последующее расширение. ООП аккумулирует лучшие идеи, воплощенные в структурном программировании и сочетает их с мощными концепциями, которые позволяют оптимально организовать программы. ООП позволяет разложить проблему на составные части. Каждая составляющая становится самостоятельным объектом, содержащим свои собственные коды и данные, относящиеся к этому объекту. Программа в целом представляет собой набор объектов и связей между ними.

ООП базируется на типе данных, называемым “*Класс*” *(Class*), и поддерживает три основные концепции: инкапсуляция, наследование и полиморфизм.

Инкапсуляция

Класс представляет собой единство трех сущностей - *полей, методов* и *свойств*. Объединение этих сущностей в единое целое и называется инкапсуляцией. Инкапсуляция позволяет во многом изолировать класс от остальных частей программы, сделать его "самодостаточным" для решения конкретной задачи. В результате класс всегда несет в себе некоторую функциональность.

Наследование

Любой класс может быть порожден от другого класса. Для этого при его объявлении указывается имя класса-родителя:

*TChildClass = **class**(TParentClass)*

Порожденный класс автоматически наследует поля, методы и свойства своего родителя и может дополнять их новыми (своими собственными). Таким образом, принцип наследования обеспечивает поэтапное создание сложных классов и возможность разработки собственных библиотек классов.

Все классы Object Pascal порождены от единственного родителя - класса *TObject*. Этот класс не имеет полей и свойств, но включает в себя методы самого общего назначения, обеспечивающие весь жизненный цикл любых объектов - от их создания до уничтожения. Программист не может создать класс, который не был бы дочерним классом *TObject*.

Принцип наследования приводит к созданию ветвящегося дерева классов, постепенно разрастающегося при перемещении от класса *TObject* к его потомкам. Каждый потомок дополняет возможности своего родителя новыми и, в свою очередь, передает их своим потомкам.

Полиморфизм

Полиморфизм - это свойство классов решать схожие по смыслу проблемы разными способами. В рамках языка Object Pascal поведение свойств класса определяется набором входящих в него методов. Изменяя алгоритм того или иного метода в потомках класса, программист может придавать этим потомкам отсутствующие у родителя специфические свойства. Для изменения метода необходимо перекрыть его в потомке, то есть объявить в потомке одноименный метод и реализовать в нем нужные действия. В результате в объекте-родителе и объекте-потомке будут действовать два одноименных метода, имеющих разную алгоритмическую основу и, следовательно, придающих объектам разные свойства. Это и называется полиморфизмом объектов.

Формализация (абстрагирование)

Основные концепции, приведенные выше, базируются на более общем принципе, называемом *формализацией (абстрагированием)*. Формальная абстракция позволяет выделить существенные для данной задачи характеристики некоторого объекта, отличающие его от всех других видов объектов и, таким образом, четко определяет его концептуальные границы. При этом выделяют следующие виды этого понятия:

- *абстракция сущности* - объект представляет собой полезную модель некой сущности в предметной области;
- *абстракция поведения* - объект состоит из обобщенного множества операций;
- *абстракция виртуальной машины* - объект группирует операции, которые либо вместе используются более высоким уровнем управления, либо сами используют некоторый набор операций более низкого уровня;
- *произвольная абстракция* - объект включает в себя набор операций, не имеющих друг с другом ничего общего.

Как уже говорилось выше, класс объединяет в себе три сущности: поля, методы и свойства.

- *Поля* - представляют собой переменные в классе и предназначены для хранения данных во время работы экземпляра класса (объекта).
- *Методы* - это процедуры и функции, используемые в пределах класса.
- *Свойства* - определяются тремя элементами: специальным полем и двумя методами, которые осуществляют доступ к этому полю (*чтение* и *запись* его значений). Другие возможности доступа к свойству отсутствуют.

Свойства необходимы для выполнения дополнительных действий при чтении из поля и/или записи в него. Например, при присвоении координатам шахматной фигуры новых значений, мы должны сначала проверить может ли фигура туда сходить, затем переместить в это место картинку, изображающую фигуру, и только потом присвоить полю, хранящему эти координаты, необходимые значения. Все эти действия можно каждый раз выполнять последователь-

но, но более рационально всю последовательность объединить в метод, осуществляющий запись в поле. Тогда каждый раз при перемещении фигуры, нам не нужно будет заботиться о том, что необходимо проверить может ли фигура сходить на новое место и просто присвоить новое значение в свойство, которое само выполнит все необходимые действия и проверки.

Постановка задачи

Игра “Шахматы” дает возможность прекрасно проиллюстрировать возможности современного программирования с использованием объектно-ориентированного подхода. Перед тем, как начать создавать программу, сформулируем основные требования, которые необходимо выполнить. Цели и задачи нашей программы будут заключаться в том, чтобы обепечить:

- создание шахматной доски с буквенно-цифровой разметкой;
- начальную расстановку шахматных фигур на доске;
- реализацию ходов всех шахматных фигур по правилам;
- возможность “срубить” фигуру по правилам;
- счет количества ходов и их очередность;
- обработку ситуаций “ШАХ”, “МАТ” и “ПАТ”.

Таким образом, эта программа не будет предназначена для игры пользователя с компьютером, то есть в нашу задачу не входит разработка каких-либо алгоритмов обработки шахматной ситуации. Эта задача является весьма сложной, и над ней многократно работали и продолжают по сей день работать целые коллективы высококласных пограммистов, когда речь идет, например, о поединках суперкомпьютера с Чемпионом Мира по шахматам.

С помощью подобной программы можно будет просто играть двум соперникам, пользуясь собственными практическими навыками шахматной игры. Реализация программы может быть самой различной в зависимости от вкуса и желания программиста, поэтому в данной статье постараемся описать проблему в самом общем виде, при этом рассмотрим несколько возможных вариантов ее создания.

Реализация: шаг первый – создание шахматной доски

Сначала создадим шахматную доску, по которой будут перемещаться фигуры. Прежде всего, необходимо решить из чего будет состоять доска, так как это будет существенно влиять не только на то, как мы ее будем отображать, но и как будут перемещаться по ней фигуры. Для конечного пользователя это не будет иметь абсолютно никакого значения, а вот для программиста это является главным и весьма существенным фактором.

Общий вид создаваемого приложения представлен на рисунке. Далее кратко опишем три возможных варианта создания шахматной доски, используя: *TPanel*, *TShape* и *TCanvas*.

Сначала определим размеры окна будущего приложения, которые следует поместить в обработчик события *OnCreate* главной формы:

```
procedure TForm1.FormCreate(Sender: TObject);
 begin
      ClientHeight:=440;
      ClientWidth:=440;
      BorderStyle:=bsSingle;
 end;
```

Здесь первые две строки отвечают за высоту и ширину клиентской части формы (окна приложения) соответственно, а третья строка запрещает изменять размеры формы. Значение 440 получилось следующим образом: 20 px — расстояние от всех краев формы до доски, где будет нанесена буквенно-цифровая разметка; 8 клеток по 50 px дадут нам еще 400 px. Итого, получим указанное значение в 440 px.

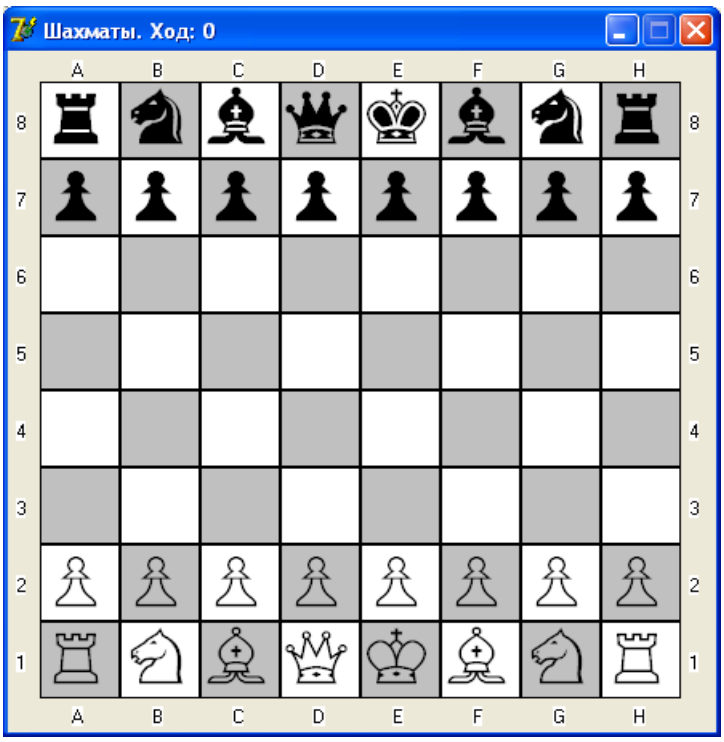

Рис. *Вид приложения «Шахматы»*

Доска как экземпляр класса TShape или TPanel

В разделе объявления глобальных переменных *VAR* или в разделе *private* главной формы (все зависит от конкретной реализации) объявим двумерный массив переменных 8 на 8 типа *TKletka* следующим образом:

DOSKA: array[1..8,1..8] of TKletka;

В дальнейшем, массив клеток доски будем обозначать именно так для обобщения всех вариантов конструирования доски. Здесь в качестве объектов типа *TKletka* (запись в общем виде) в конкретной реализации программы можно использовать объекты типа *TShape* (массив фигур-квадратов) или объекты типа *TPanel* (массив панелей). Как раз в этом различий мало: несложно доску из одного вида объектов быстро «превратить» в доску из другого вида объектов и наоборот.

Отрисовка доски будет происходить также при событии *OnCreate* главной формы, куда следует поместить следующий код:

```
//---------------- Создание шахматной доски -------------------------
for i:=1 to 8 do
  for j:=1 to 8 do
//Двойной цикл обеспечит нам квадрат из клеток 8 на 8
   begin
     DOSKA[i,j]:=TKletka.Create(Form1);
//Вызов конструктора класса для выделения памяти под объект
     DOSKA[i,j].Parent:=Form1;
//Определяем объект-родитель, в который будут помещаться клетки доски:
//поскольку размещение на главной форме, ее и указываем
     DOSKA[i,j].Height:=50;
     DOSKA[i,j].Width:=50;
//Эти две строки отвечают за размеры клеток: высота и ширина
     DOSKA[i,j].Left:=20+50*(i-1);
     DOSKA[i,j].Top:=20+50*(j-1);
//Здесь определяется положение клеток:
//Left - расстояние от левого края формы до объекта,
//Top - от верхнего края формы до объекта.
//Вычисление координат происходит по следующему принципу:
//ширина или высота клетки (соответственно для вычисления расстояния
//от левого края или от вехнего) умножается на номер клетки за вычетом
//единицы (т.е. начиная с нуля) и добавляется расстояние от края доски
//до доски в целом. В нашем примере это расстояние равняется 20, чего
//как раз хватит для написания буквенно-цифровых меток
     if odd(i+j) then DOSKA[i,j].Цвет:=clSilver
                 else DOSKA[i,j].Цвет:=clWhite;
//Здесь клеткам присваивается цвет. Для чередования цвета используется
//функция odd(X:integer): boolean, которая определяет четность
//аргумента и в случае четности возвращает значение true
     DOSKA[i,j].OnDragDrop:=Form1.OnImageDrop; //ЗАКОММЕНТИРОВАТЬ!
```

```
    DOSKA[i,j].OnDragOver:=Form1.OnImageOver; //ЗАКОММЕНТИРОВАТЬ!
//Присваивание событиям OnDragDrop и OnDragOver
//класса TKletka наших методов в качестве обработчиков этих событий
   end;
//---------------------------------------------------------------------
```

Все, доска готова. Можно запустить программу и посмотреть корректно ли она отобразилась. При запуске следует закомментировать две строчки событий Drag&Drop, поскольку они пока еще не реализованы. Если все набрано правильно, то на форме отобразится доска 8 на 8, причем левый нижний квадрат должен быть черным.

Для полной картины следует подписать клетки буквами от A до H по горизотнали и цифрами от 1 до 8 по вертикали (см. рис). Для этого в разделе объявления глобальных переменных или в разделе *private* главной формы объявим двумерный массив меток:

```
POLE: array[1..4,1..8] of TLabel;
```

Таким образом, мы объявили переменные для 32-х меток - 16 из них для букв сверху и снизу доски, а другие 16 - для цифр слева и справа. Ниже приведен еще один фрагмент кода события *OnCreate* главной формы, который позволит сделать на доске необходимые надписи:

```
//---------------------------------------------------------------------
 for i:=1 to 4 do
  for j:=1 to 8 do
    begin
     POLE[i,j]:=TLabel.Create(Form1);    //Создание метки
     POLE[i,j].Parent:=Form1;            //на форме (родитель)
     POLE[i,j].Font.Style:=[fsBold];     //Стиль шрифта
     POLE[i,j].Transparent:=true;        //Прозрачная метка
     with POLE[i,j] do
      case i of
       1: begin Left:=40+(j-1)*50; Top:=5;
```

```
        Caption:=chr(64+j);   end;  //Верхняя строка (A-H)
      2: begin Left:=5;          Top:=Form1.ClientHeight-j*50;
        Caption:=IntToStr(j); end;  //Левый столбец (1-8)
      3: begin Left:=40+(j-1)*50; Top:=425;
        Caption:=chr(64+j);   end;  //Нижняя строка (A-H)
      4: begin Left:=425;        Top:=Form1.ClientHeight-j*50;
        Caption:=IntToStr(j); end;  //Правый столбец (1-8)
      end; {Case}
    end;
//---------------------------------------------------------------------
```

Теперь можно запустить программу еще раз и проверить, как выглядит общий вид шахматной доски в окончательном виде с нанесенной разметкой.

Доска как экземпляр класса TCanvas

В этом случае все зависит от фантазии программиста, так как вариантов сделать доску в этом случае существует великое множество. Например, в обработчике события *OnPaint* главной формы можно написать следующий код, который опирается на те же цифровые значения, что и ранее:

```
procedure TForm1.FormPaint(Sender: TObject);
var i,j: byte;
begin
for j:=1 to 8 do
  for i:=1 to 8 do
    begin
      if odd(i+j) then Canvas.Brush.Color:=clSilver
        else Canvas.Brush.Color:=clWhite;
    Canvas.Rectangle(20+50*(i-1),20+50*(j-1),70+50*(i-1),70+50*(j-1));
    end;
end;
```

Как и ранее, размеры окна приложения должны быть заданы в событии *OnCreate* главной формы:

```
procedure TForm1.FormCreate(Sender: TObject);
 begin
    ClientHeight:=440;
    ClientWidth:=440;
    BorderStyle:=bsSingle;
 end;
```

Таким образом, мы рассмотрели в качестве примера построение шахматной доски в трех различных вариантах. Можно использовать что-то похожее или написать что-то абсолютно свое. Особенно творчески можно подойти к рисованию доски на свойстве *Canvas* формы, так как она представляет из себя чистый холст для рисования и предлагает обширные возможности для использования. При изображении шахматных фигур в этом случае можно воспользоваться специальными ”шахматными” шрифтами (шрифты с изображениями готовых шахматных фигур), которые можно найти, например, в Интернете.

Не смотря на то, что создание доски во всех трех вышеописанных вариантах очень похоже, реализация перемещения фигур по ним абсолютно разная. Далее мы рассмотрим, в каких случаях, какие варианты возможны.

Реализация: шаг второй – проектирование класса TFigura

На данном этапе нашей задачей является правильно спроектировать класс *TFigura,* так как в этом случае можно будет легко разобраться в коде программы, а в случае необходимости, быстро найти ошибку, что-то исправить или дополнить. Для этого необходимо выделить основные характеристики шахматной фигуры как объекта, которые и будут описывать наш класс. Такой метод, как уже упоминалось выше, называется *абстракцией*. Итак, определим основные характеристики шахматных фигур и операции над ними, которые нам необходимо реализовать в проекте.

Во-первых, сначала необходимо выбрать “класс-родитель”, чтобы наследовать максимум необходимых характеристик (сущностей) и методов.

- Если доска реализована объектами типа *TShape* или *TPanel*, на которые будут помещаться изображения шахматных фигур, то в этом случае в качестве родительского класса следует выбрать класс *TImage*.
- Если же доска реализована объектом типа *TCanvas*, по которой мы собираемся писать “шахматным” шрифтом, в качестве родительского класса целесообразно выбрать класс *TLabel*.

Конечно, возможны и другие варианты, например: ничто не мешает использовать объекты типа *TLabel* и для первого случая реализации шахматной доски.

Во-вторых, чтобы отобразить фигуры на экране, расставить их на доске и присвоить всем полям начальные значения, нам необходим *конструктор*. Варианты его реализации рассмотрим несколько позже.

В-третьих, необходимо хранить координаты фигур (начальные - до совершения хода и конечные – после совершения хода), которые нужно реализовать свойствами. Следовательно, должны присутствовать методы записи и чтения в эти свойства непосредственно из полей. Поскольку в шахматах имеется только два цвета фигур (белые и черные) его лучше реализовать не в виде свойства, а в виде логической переменной, которая принимает значение *true* для белых фигур и *false* для черных. Хотя по желанию программиста цвет фигур можно также оформить в виде свойства.

В-четвертых, все фигуры должны ходить по правилам, следовательно, должен существовать *метод, определяющий правило хода* для каждой фигуры. Так как в шахматах существует 6 разновидностей фигур (Король, Пешка, Ферзь, Слон, Конь и Ладья), нужно объявить у нашего класса *TFigura* шесть классов-потомков (подклассов) и сделать метод правила хода *полиморфным*. Назовем его *HOD*.

В-пятых, чтобы фигуры не могли прыгать друг через друга, необходим метод, выполняющий соответствующую проверку. Назовем его *JUMP* - *метод прыжка*. Здесь исключение составляют Конь (ему разрешено прыгать через фи-

гуры) и Король (он вообще “не умеет прыгать” и его перемещение в любую сторону состаляет только одну клетку).

В-шестых, у фигур на основании двух предыдущих методов (ход по правилам с учетом занятых клеток на пути фигуры) должен присутствовать метод перемещения на новое место – *правильный ход*. Он, в свою очередь, может быть реализован двумя разными способами: на пустую клетку и на занятую. В последнем случае следует либо “срубить” фигуру, если она противоположного цвета, либо запретить ход, если он сделан на фигуру своего цвета.

В-седьмых, следует реализовать обработку ситуации “Шах”, а при полном рассмотрении также обработку ситуаций “Мат” и “Пат”.

Все остальное зависит от конкретной реализации. По желанию программиста и от конкретного проектирования к этому списку может быть добавлен ряд дополнительных свойств и полей. Типы всех полей, используемых в классе, определяются только программистом и могут быть самыми различными. Например, координаты фигур X,Y можно записать как двумя полями типа *byte*, так и использавать стандарный тип *TPoint* или создать собственную запись, состоящую из двух собственных типов и т.д. Вариантов множество, поэтому далее для обозначения всех типов будут использовны условные обозначения. Исходя из вышеописанных рассуждений, наш класс в общем виде можно описать следующим образом:

```
//---------- Общее описание класса TFigura -------------------------
TFigura = class(TРодитель)
 private
   FPlace: TПоложение;
   FColor: TЦвет;       //Можно использовать тип Boolean
   procedure SetPlace(Значение: TПоложение);
                  //Метод перемещения фигуры на новое место
   procedure SetColor(Значение: TЦвет);
        //Метод смены цвета фигуры (необязателен, если Boolean)
   property MESTO: TПоложение read FPlace write SetPlace;
```

```
        //Свойство, отвечающее за положение фигуры на доске
    property Color: TЦвет read FColor write SetColor;
        //Свойство, отвечающее за цвет фигуры
        //(необязательно, если Boolean)
    function HOD(Координаты: TПоложение): boolean; virtual; abstract;
                //Метод, определяющий правило хода фигуры
    function JUMP(Координаты: TПоложение): boolean;
        //Метод проверки на отсутствие занятых клеток на пути хода
    constructor ConstructorName(Координаты: TПоложение; Цвет: TЦвет);
                //Конструктор фигуры
  end;
//-------------------------------------------------------------------
```

Далее, создадим шесть классов-потомков (подклассов). Для каждого типа фигур - свой подкласс, в котором переопределен метод *HOD*, где описывается правило хода для каждой конкретной фигуры. При правильном ходе фигуры функция *HOD* получает значение *true*.

```
//-------------------------------------------------------------------
TPawn = class(TFigura)    //Пешка
  private
    function HOD(XY: TПоложение): boolean; override;
  end;
TRook = class(TFigura)    //Ладья
  private
    function HOD(XY: TПоложение): boolean; override;
  end;
TKing = class(TFigura)    //Король
  private
    function HOD(XY: TПоложение): boolean; override;
  end;
TBishop = class(TFigura)  //Слон
```

```
  private
    function HOD(XY: TПоложение): boolean; override;
  end;
TQueen = class(TFigura)   //Ферзь
  private
    function HOD(XY: TПоложение): boolean; override;
  end;
TKnight = class(TFigura)  //Конь
  private
    function HOD(XY: TПоложение): boolean; override;
  end;
//---------------------------------------------------------------
```

Реализация: шаг третий – разработка методов класса TFigura

Нам понадобятся следующие глобальные переменные: массив из тридцати двух элементов для хранения всех фигур и логическая переменная для определения очередности хода, в которой будем хранить цвет фигур, чья очередь хода. Кроме того, нужен счетчик ходов. Опишем их в разделе объявления глобальных переменных следующим образом:

```
Var Figures: array[1..32] of TFigura; //Массив фигур

OCHERED: boolean;              //Очередность хода
XODOV: byte=0;                 //Счетчик ходов
```

Конструктор

Теперь переходим к описанию методов. Самый важный из них – конструктор, главной задачей которого является выделение памяти под объект. Непосредственной работой с памятью занимается среда разработки, то есть сама среда Delphi и к обязанностям программиста относится просто вызов необходимых конструкторов для нужных объектов. Также в конструкторе определя-

ются начальные значения для полей класса и изменяются значения необходимых свойств у создаваемых объектов.

Картинки с изображением фигур будут загружаться из специальной папки с набором нужных изображений или предварительно созданного файла ресурса. Названия картинок нужно строить по определенным правилам: *цвет_ИмяКласса*, например: *black_TPawn*, *white_TKing* и т.д. Для использования файла-ресурса, в котором хранятся изображения фигур, такой файл следует подключить специальной директивой *{$R Фигуры.res}*.

Ниже приведен пример возможного общего вида конструктора для класса *TFigura*, порожденного от класса *TImage*.

```
//---------- Общий вид конструктора фигуры -------------------------
constructor TFigura.ConstructorName(Координаты: TПоложение,
                                    Цвет: boolean);
  begin
    inherited Create(Form1); //Директива inherited используется
    Parent:=Form1;           //в зависимости от реализации
//Указываем объект-контейнер, на который помещается создаваемый объект
//в нашем случае  - это форма (или объект класса TKletka,
//если доска состоит из таких объектов)
    FPlace:=Координаты; //Положение фигуры на доске
    MESTO:=FPlace;      //Сохранение координат в свойстве для работы
    FColor:=Цвет;       //Цвет фигуры
//-------- НАСТРОЙКА IMAGE (ФИГУРЫ) --------------------------------
    Transparent:=true;  //Прозрачность фона картинки
    Width:=40;          //Высота картинки
    Height:=40;         //Ширина картинки
//---------- Пример загрузки картинок из ресурса -------------------
if Цвет then
   Picture.Bitmap.Bitmap.LoadFromResourceName(HInstance,'white_'
                        + Self.ClassName+'.bmp');
```

```
      else
  Picture. Bitmap.LoadFromResourceName(HInstance,'black_'
                    + Self.ClassName+'.bmp');
//---------------------------------------------------------------
   DragMode:=dmAutomatic;    //Автоматическая обработка Drag&Drop
//Событие Drag&Drop разделено на два:
   OnDragOver:=Form1.OnImageOver;
   OnDragDrop:=Form1.OnImageDrop;
//DragOver - отвечает за перенос объекта
//DragDrop – отвечает за бросание его на другой объект
end;
//---------------------------------------------------------------
```

Так как при вызове конструктора объект только создается, то для дальнейшей работы с ним, этот объект нужно поместить в компонент массива *Figures[i]* соответствующего типа. Например, поместим белую пешку в качестве первого элемента нашего массива на доску в клетку [1,2]:

```
Figures[1]:=TPawn.Create(1,2,true);
```

Разумеется, в зависимости от типа координат, входные параметры могут отличаться: здесь приведен только общий пример.

В данной записи (создание пешки) вызывается конструктор класса *TPawn*. Так как конструктор для этого класса в нашем случае не реализован, то будет вызываться конструктор ближайшего родителя, то есть класса *TFigura*, в котором создается картинка, в нее в соответствии с цветом фигуры загружается нужное изображение и помещается на доску по указанным координатам. Для последующего обращения к этой пешке, она сохраняется в компоненте массива *Figures[1]*. Теперь к полям, методам и свойствам этого объекта можно обращаться следующим образом:

Figures[1].ИмяМетода;

Например, переместить фигуру на другую клетку можно следующим образом:

Figures[1].HOD(Координаты);

Описание методов записи в свойства здесь приводить не будем, так как их содержание может очень сильно различаться, уточним лишь то, что там необходимо присвоить вводимые координаты в поле положения фигуры, и перемещать в них картинку. Они будут рассмотрены далее. Будем считать, что для исходной шахматной позиции фигуры расставлены.

Приведем еще один пример возможного общего вида конструктора для класса *TFigura*, порожденного уже от класса *TLabel.* Он позволит работать с упямянутыми выше готовыми “шахматными” шрифтами. Для этого необходимо в разделе *Uses* программного модуля Unit1 подключить специальную библиотеку работы со шрифтами *STRUTILS*. Ниже представлен пример кода, реализующий такой конструктор:

```
//================== Создание фигуры (Конструктор) ==================

constructor TFigura.FigCreate(Y,X: byte; Color: boolean);

 begin

  Create(Form1);     // Выделение памяти под объект

  Parent:=Form1;     // Указание на родителя

  FPlace.Y:=Y;       // СТОЛБЕЦ

  FPlace.X:=X;       // СТРОКА

  MESTO:=FPlace;     // Сохраняем координаты

  FColor:=Color;     // Цвет фигуры (true-белые, false-черные)

  //------------ НАСТРОЙКИ LABEL (ФИГУРЫ) --------------------

  Font.Name:='Chess Alpha'; // НАЗВАНИЕ ШРИФТА

  Font.Color:=clNavy;       // ЦВЕТ ФИГУР
```

```
Font.Size:=36;          // РАЗМЕР ШРИФТА
Transparent:=true;      // ПРОЗРАЧНОСТЬ
AutoSize:=false;        // ОТКЛЮЧАЕМ АВТОРАЗМЕР
Height:=50;             // ВЫСОТА
Width:=50;              // ШИРИНА
//---------- ЗАГРУЗКА КАРТИНОК (ШРИФТ) --------------------
if Color then
  case AnsiIndexText(ClassName,   //БЕЛЫЕ (Color=true)
    ['TPeshka','TRook','TKnight','TBishop','TFerz','TKorol']) of
      0: Caption:=#112;   // ПЕШКА
      1: Caption:=#114;   // ЛАДЬЯ
      2: Caption:=#104;   // КОНЬ
      3: Caption:=#98;    // СЛОН
      4: Caption:=#113;   // ФЕРЗЬ
      5: Caption:=#107;   // КОРОЛЬ
  end else
  case AnsiIndexText(ClassName,   //ЧЕРНЫЕ (Color=false)
    ['TPeshka','TRook','TKnight','TBishop','TFerz','TKorol']) of
      0: Caption:=#111;   // ПЕШКА
      1: Caption:=#116;   // ЛАДЬЯ
      2: Caption:=#106;   // КОНЬ
```

```
        3: Caption:=#110;   // СЛОН
        4: Caption:=#119;   // ФЕРЗЬ
        5: Caption:=#108;   // КОРОЛЬ
    end;
  //---------------- ТАСКАЛКА ------------------------------
  DragMode:=dmAutomatic;      //Режим автоперетаскивания фигуры
  OnDragOver:=Form1.FormDragOver;  //Перетаскивание фигуры
  OnDragDrop:=Form1.FormDragDrop;  //Опускание фигуры на форму
end;
```

Итак, мы рассмотрели два возможных варианта реализации конструктора шахматной фигуры.

Метод HOD (Правило хода фигуры)

Следующее, что необходимо сделать – это научить фигуры правильно ходить и не перепрыгивать друг через друга. Следовательно, сначала необходимо написать эти методы. Правило хода фигуры – полиморфный метод, то есть каждый подкласс описывает его по-своему, поэтому в классе *TFigura* мы объявили его как виртуальный метод (*virtual*). Добавленное слово *abstract* означает, что этот метод используется только в классах-потомках (подклассах), и в основном классе его реализация отсутствует.

Наш метод проверки хода должен возвращать *true*, если ход возможен и *false*, если он не по правилам. В качестве параметров, мы должны передавать в наш метод координаты нового места хода (конечной клетки), так как начальные координаты можно взять из свойства, в котором они хранятся. Другими словами, это будет функция логического типа с одним входным параметром типа *ТПоложение*, в котором будем передавать конечные координаты.

Рассмотрим создание метода правила хода на примере фигуры «Ферзь», который имеет право ходить по вертикалям, горизонталям и диагоналям. Представим конечную координату как *Ferz_XY*. Чтобы можно было понять, правильно ли был выполнен ход, необходимо сравнить координаты по следующим признакам:

- если ход произведен по вертикали, то у начальной и конечной координат совпадает координата X;
- если ход произведен по горизонтали, то у начальной и конечной координат совпадает координата Y;
- если ход произведен по диагонали, то модуль изменения координат по X равен модулю изменения координат по Y.

На языке Object Pascal это можно описать следующим образом:

```
function TQueen.HOD(Ferz_XY: TПоложение): boolean;
 begin
  Result:=((MESTO.X=FERZ_XY.X) or (MESTO.Y=FERZ_XY.Y) or
       (abs(MESTO.X-FERZ_XY.X)=abs(MESTO.Y-FERZ_XY.Y)));
 end;
```

В данном фрагменте записи MESTO.X и MESTO.Y отражают начальные, а FERZ_XY.X и FERZ_XY.Y конечные координаты фигуры соответственно. Функция дает значение true, если ход выполнен верно и false - в противном случае.

Исходя из данного примера, несложно написать проверки для фигур: Ладья, Слон и Король. Конь ходит на две клетки по вертикали и на одну по горизонтали или наоборот: две по горизонтали и одна по вертикали. Оба этих случая необходимо соединить логическим оператором «или».

Самой сложной в написании этого метода является пешка. Пешки ходят только вперед, то есть белая пешка ходит только вверх, а черная – только вниз и только на одну клетку. Если же они находятся на «своей» горизонтали (седьмой для черных и второй для белых), то сходить можно на две клетки; ходят

они только прямо, а рубят только по диагонали. Это все надо учесть в реализации хода пешки.

Метод JUMP (Проверка свободного пути)

Для того, чтобы фигуры могли правильно ходить, необходимо еще выполнить проверку на наличие занятых клеток на пути хода фигуры и, в случае положительного результата, запрещать ход.

Данный метод, аналогично предыдущему, должен возвращать логическое значение: *true* – если ход возможен, *false* – в противном случае. Как и в предыдущем случае, метод будет принимать значение конечной координаты, а начальную - брать из свойства. Сейчас рассмотрим, как будет работать метод. Он должен найти фигуру, которая расположена между начальной и конечной клетками движения. Если такая “фигура-барьер” не найдена, то ход разрешен. Выполнять проверки необходимо в трех случаях:

- ход совершен по горизонтали;
- ход совершен по вертикали;
- ход совершен по диагонали.

Под эти три случая не попадает только Конь, а для него такая проверка не требуется, так как он может прыгать через другие фигуры. Все эти случаи описываются совершенно одинаково, поэтому рассмотрим только один из них – вертикаль.

```
//-------------------------------------------------------------------
function TFigura.JUMP(Координаты: TПоложение): boolean;
var i: byte;
  begin
   Result:=true;
//Эта строка необходима для начального присваивания значения
//результату функции. Таким образом, мы изначально разрешаем ход,
//а в случае найденной занятой клетки – запрещаем.
  for i:=1 to 32 do          //Проверка всех фигур
```

```
    if Figures[i]<> nil then  //если фигура есть на доске
      begin
      //---------- Проверка по вертикали ----------------------
      //ищем фигуру-барьер на пути хода,
      //координата X которой совпадает с нашей
      if (Координаты.X = MESTO.X) and
        (Figures[i].MESTO.X = MESTO.X) and
      //если такая фигура найдена, то проверяем
      //находится ли она на промежутке движения нашей фигуры
       (((Figures[i].MESTO.Y > MESTO.Y) and
         (Figures[i].MESTO.Y < Координаты.Y)) or
        ((Figures[i].MESTO.Y < MESTO.Y) and
         (Figures[i].MESTO.Y > Координаты.Y)))
          then Result:=false;
          //если и это условие выполнено,
          //то занятая клетка найдена – ход запрещен
       end;
//-----------------------------------------------------------------
```

Аналогичным образом проверяются горизонталь и диагональ. Три таких блока дают нам рабочий метод проверки свободного пути.

Метод SETPLACE (Перемещение фигуры на новое место)

Данный метод производит изменение положения фигуры на доске и существенно зависит от конкретной реализации. Ниже приведены фрагменты кода для случаев конструирования доски с помощью *TPanel и TShape:*

```
//-----------------------------------------------------------------
  procedure TFigura.SetPlace(NEW_XY: TПоложение);
  begin
    FPlace:=NEW_XY;
    Parent:=Form1.DOSKA[NEW_XY.X, NEW_XY.Y]; //для TPanel
```

```
   LEFT:=DOSKA[NEW_XY.X, NEW_XY.Y].Left-5;  //для TShape(Столбец)
   TOP:= DOSKA[NEW_XY.X, NEW_XY.Y].Top-5;   //для TShape(Строка)
 end;
```

Фрагмент кода для случая конструирования доски с помощью *TCanvas*:

```
//---------------------------------------------------------------
   procedure TFigura.SetPlace(NEW_XY: TПоложение);
     begin
       FPlace:=NEW_XY;
       Top:= (8-NEW_XY.Y)*50+20;       //СТРОКА
       Left:=(NEW_XY.X-1)*50+20;       //СТОЛБЕЦ
     end;
//---------------------------------------------------------------
```

Метод SHAX (Проверка на шах)

Сейчас создадим метод, который выполняет проверку на “шах”. Следует отметить, что функция этого метода может быть описана и как метод класса *TFigura* и как независимая функция. Все, как всегда, зависит от желания программиста. Можно предложить несколько способов ее реализации.

Ниже рассмотрен случай, когда данный метод должен искать фигуру противоположного цвета тому, который указан в параметре данной функции. При этом искомая фигура может сходить на короля, цвет которого обозначен тем же параметром. Другими словами, метод должен проверить каждую фигуру, которая в данный момент присутствует на доске и, если ее цвет не совпадает с цветом атакуемого Короля, узнать, выполняются ли для нее условия методов *HOD* и *JUMP* одновременно на “вражеском королевском месте”. Если оба метода одновременно вернули значение *true*, значит и метод *SHAX* должен вернуть в качестве результата значение *true*.

```
//----------------- СИТУАЦИЯ <ШАХ> --------------------------------
```

```
function SHAX(Color: TColor): boolean;
var i,King: byte;
  begin
    Result:=false;
    //Автопоиск необходимого короля
    for i:=1 to 32 do
       if (Figures[i] is TKing) and (Figures[i].FColor<>Color)
          then King:=i; //Определили номер атакуемого Короля
    //Проверка выполняемости всех описанных выше условий
    for i:=1 to 32 do
     if Figures[i]<> nil then
      if (Figures[i].FColor<>Figures[King].FColor)
        and Figures[i].HOD(Figures[King].MESTO)
        and Figures[i].JUMP(Figures[King].MESTO) then
       begin
        ShowMessage('ШАХ!');
        Result:=true;
       end;
  end;
//-----------------------------------------------------------------
```

Итак, мы реализовали метод, организующий обработку ситуации “Шах”. Его достаточно просто можно доработать таким образом, чтобы в сообщении указывалось, какому именно Королю (белому или черному) объявлен “Шах”.

Методы перемещения фигур по доске Drag&Drop

Теперь все подготовлено к тому, чтобы заставить наши фигуры правильно перемещаться по шахматной доске. Создадим соответствующие методы для перетаскивания фигур *Drag&Drop*, тем более, что они уже были использованы нами в конструкторе, но их реализация пока отсутствует.

В нашем проекте изначально отсутствуют объекты типа *TImage*, в которых хранятся изображения шахматных фигур, поскольку они создаются в процессе работы программы динамически. А ведь именно их и предстоит претаскивать! Как же создать метод для еще несуществующего объекта?

Для того чтобы создать метод какого-либо события, необходимо использовать обработчик такого события. Для этого необходимо вызвать нужное событие из Инспектора Объектов у данного объекта или у любого другого, у которого есть такое событие. Например, можно найти в Инспекторе Объектов события *OnDragDrop* и *OnDragOver* для формы и щелкнутье дважды мышью в соответствующих пустых полях. Среда программирования Delphi сама создаст пустые обработчики для этих событий. После чего нужно скопировать *списки параметров* появившихся методов в свои собственные методы, которые мы разрабатываем. В нашем примере это методы *OnImageDrop* и *OnImageOver*, которыми необходимо дополнить класс формы (для наглядности в приведенном ниже фрагменте кода они подчеркнуты):

```
//---------------------------------------------------------------------
TForm1 = class(TForm)
   procedure FormCreate(Sender: TObject);
   procedure OnImageDrop(Sender, Source: TObject; X,Y: Integer);
   procedure OnImageOver(Sender, Source: TObject; X,Y: Integer;
                  State: TDragState; var Accept: Boolean);
 private
  { Private declarations }
 public
  { Public declarations }
 end;
//---------------------------------------------------------------------
```

Для того чтобы эти методы обрабатывали необходимые события, нужно присвоить их этим событиям, как это мы делали ранее при создании доски:

DOSKA[i,j].OnDragDrop:=Form1.OnImageDrop;

DOSKA[i,j].OnDragOver:=Form1.OnImageOver;

В этом случае ни список параметров, ни скобки уже не указываются. В условных обозначениях это можно еще написать следующим образом:

ЭКЗЕМПЛЯР_ОБЪЕКТА.СОБЫТИЕ:=МЕТОД;

Напомним, что данные методы здесь использованы для случая, когда в качестве родительского класса выбран класс *TImage*. Теперь кратко опишем содержание самих этих методов.

Метод OnImageOver (Перемещение объекта).

Здесь нужно написать только одну строчку:

Accept:=true;

Эта запись разрешает использование механизма перетаскивания *Drag&Drop*, то есть одному объекту (доске) разрешено принять другой объект (фигуру) при отжатии правой клавиши мыши, если один объект находится над другим.

Метод OnDragDrop (Опустить объект).

Здесь нужно описать действия, которые необходимо выполнить после перемещения фигуры на доске на новое место. Как уже говорилось выше, существует два варианта событий: перемещение фигуры на пустую клетку или на занятую. Во втором случае, если клетка занята фигурой соперника, то ее необходимо “срубить”, если союзной – запретить перемещение. Ниже представлен сокращенный вариант кода одного из возможных обработчиков событий, записанный в общем виде:

```
//-------------------------------------------------------------
procedure TForm1.OnImageDrop(Sender,Source: TObject; X,Y: Integer);
var NewPlace: TПоложение; //Новые (конечные) координаты фигуры
   OldPlace: TПоложение; //Исходные (начальные) координаты фигуры
                  //(используются в случае отмены хода)
```

```
   SRUBLENA: byte;  //Номер срубленной фигуры в массиве Figures[i]
        ...
begin
 //Проверка куда совершен ход: на пустую клетку или на занятую
 if Sender is TKletka then
  begin
  //TKletka– имя класса, экземпляром которого является клетка доски
  //далее, вычисляем координаты этой клетки
  NewPlace.X:=((TKletka(Sender).Left -20) div 50)+1;
  NewPlace.Y:=8-((TKletka(Sender).Top -20) div 50);
  //Проверка, возможен ли ход на эту клетку: выполняемость
  //условий методов HOD и JUMP при правильной очередности хода
 if TFigura(Source).HOD(NewPlace) and TFigura(Source).JUMP(NewPlace)
   and (TFigura(Source).FColor=OCHERED) then
     begin
        OldPlace:=TFigura(Source).MESTO;
        TFigura(Source).MESTO:=NewPlace;
        if SHAX(OCHERED) then TFigura(Source).MESTO:=OldPlace;
                //Возврат фигуры на старое место при Шахе
         ...
     end;
        OCHERED:=not OCHERED //Переход хода
        XODOV:=XODOV+1;     //Увеличение показаний счетчика ходов
         ...
//-----------------------------------------------------------------
```

Остается описать вариант хода с рубкой фигуры противника. Этот случай выявляется после проверки условия *if Sender is TFigura then*, аналогичной проверке хода на пустую клетку, поэтому приводить пример здесь не будем.

Метод “Сборка мусора”

В заключении рассмотрим еще один обработчик событий *OnClose*, который при закрытии окна приложения (главной формы) позволит удалить весь “мусор” и очистить оперативную память:

```
//---------------------------------------------------------------------
  procedure TForm1.FormClose(Sender: TObject;
                             var Action: TCloseAction);
  var i: byte;
  begin
    for i:=1 to 32 do FreeAndNil(Figures[i]);
  end;
//---------------------------------------------------------------------
```

Таким образом, в данной статье в общем виде рассмотрены основы создания программы “Шахматы” с использованием современной технологии объектно-ориентированного программирования.

6. ОБУЧЕНИЕ ПРОГРАММИРОВАНИЮ СТУДЕНТОВ НА ОСНОВЕ МЕТОДОЛОГИИ УНИФИЦИРОВАННОГО ПРОЦЕССА РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Текущее состояние индустрии программного обеспечения порождает потребность в ИТ-специалистах, подготовленных к применению современных технологий и подходов разработки программного обеспечения в своей профессиональной деятельности. Процесс подготовки таких специалистов должен включать изучение студентами как алгоритмизации и программирования, так и современных технологических процессов и средств программной инженерии. Такие средства позволяют обеспечить планирование процесса разработки про-

граммного проекта, облегчить описание его структурных и поведенческих моделей, а также способствовать быстрому и точному обмену проектной документацией внутри группы разработчиков.

В настоящее время существует множество подходов к организации жизненного цикла программного обеспечения. Любая современная организация-разработчик ПО предусматривает определенную последовательность этапов развития программного проекта. При этом общая модель технологического процесса подчиняется строгой закономерности, несмотря на индивидуальные особенности планирования процесса разработки ПО в каждой такой организации.

Одной из наиболее распространенных методологий разработки ПО является «Унифицированный процесс разработки программного обеспечения» («The Unified Software Development Process», [2]). Основными приоритетами этой методологии являются так называемые «варианты использования» разрабатываемого программного продукта (в том числе, планируемый функционал и интерфейсы), его архитектура, а также экономические факторы, связанные с процессом разработки и внедрения. Эффективность методологии подтверждена многолетним опытом ее реализации в различных организациях, среди которых можно выделить такие крупные компании, как IBM, Rational Software, Oracle, Eclipse Foundation и др.

При всем многообразии путей реализации унифицированного процесса, основа методологии остается неизменной и включает в себя четыре базовые фазы: «Начало», «Уточнение», «Построение» и «Внедрение».

Другой особенностью методологии унифицированного процесса является графический метапредметный язык объектного моделирования, получивший название UML (Unified Modeling Language, унифицированный язык моделирования). Данный язык позволяет наглядно представить структурную и поведенческую модель проекта, что в совокупности со строгим описанием компонентов, их отношений и ограничений позволяет использовать его в качестве инструмента для взаимодействия участников процесса разработки. Метапредмет-

ность языка UML дает возможность его использования также и в областях, не связанных с разработкой ПО.

Данные особенности методологии унифицированного процесса разработки программного обеспечения привели к росту её популярности и принятию во множестве организаций, ориентированных на разработку и сопровождение программного обеспечения. Распространенность методологии позволяет сделать вывод о целесообразности обучения ИТ-специалистов работе в условиях реализации унифицированного процесса.

Для целенаправленного обучения программированию с использованием подходов унифицированного процесса необходимо представить теоретический материал в соответствии с последовательностью этапов разработки программного проекта. Это позволит формулировать практические задания в форме, максимально приближенной к реальным условиям разработки программного обеспечения. Ввиду того, что разработка крупных программных проектов осуществляется группами разработчиков, требуется предусмотреть возможность построения практических заданий с учетом коллективного характера деятельности. Реализация данного подхода в Уральском государственном педагогическом университете потребовала создания методической системы, учитывающей технологические особенности унифицированного процесса разработки ПО, такие как подготовка проектной документации, формирование подгрупп разработчиков, применение средств управления версиями и др. Внедрение методической системы обучения осуществлялось в течение пяти лет в рамках специализации «Компьютерные игровые технологии в образовании» при подготовке студентов по специальности «Информатика», а также при обучении программированию студентов других специальностей.

В основу методической системы легли следующие положения:

- использование методологии унифицированного процесса на протяжении всего курса обучения программированию;
- реализация метода проектов;

- организация практических заданий для небольших (3-4 человека) групп студентов;
- выбор метапредметного направления разработки программного проекта.

Разработанная в соответствии с данными положениями методическая система предполагает порядок изучения программирования, основанный на технологическом процессе разработки крупных программных проектов, включающих в себя функциональные компоненты различного назначения.

В качестве предметной области для практической составляющей методической системы были выбраны компьютерные игры. Данный выбор обусловлен тем, что игровые программы представляют собой один из наиболее сложных в разработке классов программного обеспечения, обладающий всеми необходимыми характеристиками для использования в качестве практического компонента (метапредметность, архитектурная сложность, коллективный характер разработки). При организации практических заданий обучаемые делятся на небольшие группы по 3-4 человека, осуществляющие работу над общим в пределах группы проектом на протяжении всего курса обучения.

В целях реализации подходов унифицированного процесса, разработанная методическая система предполагает разбиение процесса обучения на несколько этапов. Количество и содержание этапов приведены в соответствие с фазами унифицированного процесса; таким образом, выделяется четыре этапа обучения. Порядок этапов и их соответствие фазам унифицированного процесса приведены на рис. 1.

Рис. 1. Этапы обучения и их соответствие фазам унифицированного процесса

Методическая система предполагает изучение курса в порядке следования фаз унифицированного процесса, допуская при этом возможность возврата на предыдущий этап (с определенными ограничениями). Переход на новый этап обучения возможен лишь при усвоении основной части учебного материала текущего этапа.

Для каждого из этапов обучения определены следующие параметры:

- входные требования, предъявляемые к уровню знаний и умений обучаемого;
- цель текущего этапа;
- содержание обучения;
- методы и формы обучения;
- формы и содержание контроля;
- результат этапа.

Результаты каждого из этапов одновременно являются входными требованиями следующего этапа.

На *стартовом этапе* специальные входные требования не предусмотрены. Целью данного этапа является формирование у будущего ИТ-специалиста

знаний и умений в области проектирования программного обеспечения и начальной подготовки технического задания.

Содержание обучения на данном этапе включает в себя изучение структуры и особенностей унифицированного процесса, его основных ориентиров и средств. Формируются группы студентов, которые будут заниматься в дальнейшем разработкой и реализацией программного проекта, общего для группы. Осуществляется выбор тематики разрабатываемой программы и формулирование ее основных характеристик. На данном этапе рассматриваются базовые принципы формирования архитектуры программы и процесс планирования разработки программного проекта.

Основным методом обучения на данном этапе являются информационные и проблемные лекции.

Для оценки результатов обучения используются формы текущего индивидуального контроля. Содержание контроля — знания в области унифицированного процесса разработки и проектирования программного обеспечения.

Результатом обучения на данном этапе является сформированность знаний в области организации унифицированного процесса разработки программного обеспечения.

После достижения этого результата, осуществляется переход к *проектирующему этапу,* цель которого — формирование знаний и умений в области объектного моделирования.

На проектирующем этапе изучаются основы объектного моделирования с использованием языка UML, а также основные принципы объектно-ориентированного программирования без учёта специфики языков программирования. Рассматриваются структурные и поведенческие модели программного обеспечения, основные алгоритмы взаимодействия программных модулей. Изучаются основные шаблоны проектирования программного обеспечения и их представление средствами языка UML. Уточняется тематика разрабатываемого группами студентов программного проекта и осуществляется дальнейшее планирование его разработки.

Содержание практической части обучения на данном этапе включает в себя разработку объектной модели будущей программы, которая содержит:

- описание вариантов использования разрабатываемой программы;
- описание внутренней структуры программы;
- описание конечных автоматов, используемых в программном проекте;
- описание последовательностей взаимодействия компонентов программы в различных вариантах использования;
- описание потоков управления и последовательности действий при создании, разрушении и использовании объектов;
- описание вычислительных алгоритмов;
- описание ограничений, накладываемых на состояние объектов, а также (при необходимости) на жизненный цикл основных объектов с привязкой ко времени.

Методы обучения, используемые на данном этапе, включают в себя информационные и проблемные лекции, имитационные упражнения, игровое проектирование, а также ситуационные методы.

Оценка результатов обучения осуществляется методами тематического текущего контроля в индивидуальной и групповой форме. Оцениваются результаты разработки архитектуры программных проектов в каждой из групп студентов.

Результатом обучения является сформированность знаний и умений в области объектного моделирования и языка UML.

Целью *реализующего этапа* является формирование знаний и умений в области программирования на языках высокого уровня.

На этом этапе осуществляется изучение языков программирования высокого уровня C и C++, а также особенности реализации различных алгоритмов на этих языках. Особое внимание уделяется модульному программированию, созданию собственных библиотек функций, возможности использования сторонних библиотек, позволяющих осуществлять работу с графическими и звуко-

выми возможностями программно-аппаратной платформы, а также реализующих наиболее часто используемые алгоритмы обработки данных. Рассматриваются методы оценки сложности алгоритмов, способы оптимизации программного кода на различных уровнях, средства отладки и профилирования. Устанавливается соответствие между структурными и поведенческими моделями языка UML и их реализацией средствами языков программирования C и C++. Осуществляется реализация функциональных компонентов программных проектов, разрабатываемых студентами. Уделяется внимание подходам и технологиям модульного тестирования, рассматриваются основные способы автоматического тестирования кода программных модулей и выявления регрессий. Изучаются подходы к созданию промежуточных версий программного кода, включающих дополнительные элементы, упрощающие отладку. Рассматриваются способы профилирования программного кода, а также подходы и методы оптимизации кода на высоком, среднем и низком уровнях. Для взаимодействия студентов внутри группы используются технологии управления версиями исходного кода программы, поддерживающие коллективную работу над проектом, такие как Git и Subversion.

Для практической реализации компонентов программы, осуществляется распределение модулей и подсистем между студентами внутри группы. Реализация компонентов програмы осуществляется на изучаемых языках (C и C++) с использованием открытых и переносимых программных интерфейсов.

Ввиду многообразия используемых программно-аппаратных платформ, непосредственное изучение языков программирования C и C++ ориентируется на использование преимущественно средств стандартной библиотеки, а также дополнительных библиотек, способных работать в различных программно-аппаратных средах. Учебные примеры и упражнения формируются таким образом, чтобы обеспечить возможность компиляции и запуска под управлением большинства современных операционных систем для различных устройств, в том числе встраиваемых и портативных.

Изучение языков программирования C и C++ включает в себя следующие темы:

- основы синтаксиса языка программирования C: типы данных, литералы, операторы, алгоритмические конструкции;
- организация простейшего консольного ввода-вывода в программах на языке C;
- форматированный вывод данных, в том числе в табличной форме;
- обнаружение ошибок ввода данных, валидация вводимых значений;
- общая организация ввода-вывода, работа с файлами, управление буферизацией ввода и вывода;
- создание собственных функций, механизмы передачи данных в функцию и получения результатов;
- использование препроцессора языка C: включение заголовочных файлов, создание собственных заголовочных файлов, директивы условной компиляции, символьные константы, макроопределения;
- создание многомодульных программ, статических и динамических библиотек;
- ручное управление процессом сборки программы: синтаксис сборочных файлов для программы make, создание сложных проектов, включающих в себя более одного исполняемого файла и библиотеки;
- использование сторонних библиотек на примере кроссплатформенной мультимедийной библиотеки SDL2 и программного интерфейса OpenGL;
- основы синтаксиса языка программирования C++: новые типы данных, изменения в типах данных и поведении алгоритмических конструкций языка C;
- пространства имен, перегрузка функций и операторов;
- стандартная библиотека языка C++, организация консольного ввода-вывода (в том числе форматированного) средствами этой библиотеки,

особенности использования;

- объектно-ориентированное программирование на языке C++: изучение структуры классов стандартной библиотеки, создание собственных классов и классовых иерархий;
- реализация основных шаблонов проектирования средствами языка программирования C++;
- параметризованные классы, функции-шаблоны, элементы обобщенного программирования;
- стандартная библиотека шаблонов STL: основные классы хранения данных, классы-контейнеры и коллекции, алгоритмы обработки коллекций, итераторы;
- разработка собственных классов хранения и итераторов;
- создание программных интерфейсов для сущностей, предоставляемых библиотеками, не поддерживающими объектно-ориентированное программирование;
- основы параллельного программирования, создание и синхронизация потоков.

Приведенный список тем при необходимости может быть расширен — например, в случае появления новых возможностей в будущих стандартах изучаемых языков программирования. Во время экспериментальной реализации данной методической системы содержание было ориентировано на использование стандарта ISO/IEC 9899:2011 для языка программирования C и ISO/IEC 14882:2011 для C++.

Рассмотрение сторонней мультимедийной библиотеки (в данном случае — SDL2) позволяет обучаемым получить необходимые знания о структуре мультимедийных приложений в целом, а также о способах их реализации средствами языков C и C++. Кроме того, это позволяет наглядно представить результаты выполнения различных вычислительных алгоритмов, лежащих в основе учебных примеров.

Изучение языков программирования C и C++ неразрывно связано с разработкой коллективного проекта (в данной методической системе это — игровые программы), в связи с чем параллельно с изучением непосредственно языков программирования требуется также получить знания о способах реализации различных компонентов проекта. В качестве основы для графического компонента предлагается использовать кроссплатформенную мультимедийную библиотеку SDL2 и программный интерфейс OpenGL.

Библиотека SDL2 позволяет обучаемым начать разработку коллективного проекта вне зависимости от выбранной целевой платформы — данная библиотека позволяет создавать программы, которые могут без изменений быть скомпилированы для работы в различных программно-аппаратных средах, включая операционные системы для настольных ПК (GNU/Linux, BSD, Mac OS X, Windows и др.), для мобильных устройств (Android, Sailfish, iOS и др.) и для некоторых игровых консолей. Это подразумевает также единообразную инициализацию OpenGL на данных устройствах, что позволяет создавать также приложения, использующие аппаратное ускорение двумерной и трехмерной графики.

Данная библиотека имеет также базовые возможности для работы со звуком. Эти возможности могут быть расширены за счет использования дополнительных библиотек, также кроссплатформенных.

Основные методы обучения на данном этапе включают в себя информационные и проблемные лекции, имитационные упражнения, ситуационные методы, тематические дискуссии.

Для оценки результатов обучения применяются формы группового и индивидуального текущего контроля. Проверяются результаты реализации компонентов программного проекта в каждой из групп студентов.

На данном этапе результатами обучения являются сформированность знаний и умений в области программирования на языках C и C++, обретение опыта коллективной разработки программ, а также рабочие версии программ, разработанных в рамках коллективных проектов.

Этап представления предполагает формирование знаний и умений в области разработки программной документации. На этом этапе также осуществляется публичная защита разработанных программных продуктов.

Основными методами обучения на данном этапе являются информационные и проблемные лекции, групповые консультации и имитационные упражнения.

Содержание включает в себя изучение средств генерации программной документации по объектной модели и исходному коду программы. Рассматриваются автоматические генераторы документации на основе внедрённых в исходный код комментариев определенного вида, такие как Doxygen, а также средства и технологии ручного создания документации, например, для встроенных систем подсказки.

Осуществляется индивидуальный тематический итоговый контроль; групповой контроль осуществляется путём публичной защиты разработанного программного продукта и его апробации методом экспертных оценок. Допускается также публикация разработанных материалов в студенческих сборниках статей.

Результатами данного этапа являются сформированность знаний и умений в области создания программной документации и обретение опыта коллективной разработки, реализации и внедрения программного обеспечения.

Оценка результатов обучения осуществляется комбинацией различных методов оценивания:

- текущая успеваемость на каждом из этапов обучения проверяется в соответствии с требованиями для этих этапов;
- общий результат, который представляет собой разработанная обучаемыми программа, оценивается методом экспертных оценок.

При оценивании конечного результата обучения учитываются особенности класса разработанного программного обеспечения. Общие критерии оценивания могут включать в себя вопросы применимости разработанного про-

граммного продукта для решения поставленной задачи, эргономические качества, производительность и др.

Для предлагаемого в данной методической системе класса программного обеспечения (игровые программы) к критериям оценивания добавляются такие вопросы, как:

- графическое оформление программы: соответствие использованных технологий общей идее программного проекта, дизайн графических элементов и т. д.;
- звуковое оформление программы: соответствие звукового сопровождения игровой атмосфере, синхронизация звуковых эффектов с игровыми событиями;
- особенности взаимодействия с пользователем: поддерживаемые устройства ввода (клавиатура, указывающие манипуляторы, игровые манипуляторы и пр.), логичность и удобство управления программой, отзывчивость управления;
- сюжетная составляющая (если применимо).

Особенностью данной методической системы является ее пригодность как для обучения студентов программированию, так и для повышения квалификации ИТ-специалистов, чья профессиональная деятельность связана с разработкой, реализацией и внедрением программного обеспечения. Основа методической системы допускает адаптацию содержания обучения к разработке различных классов программных продуктов, включая системы имитационного моделирования и симуляции, системы управления производственными процессами, сложные веб-приложения и порталы, и т. д. Содержание обучения языкам программирования может быть адаптировано для изучения различных языков программирования, таких, как Java, Ruby, Python и др. Поскольку в основе структуры содержания методической системы делается акцент на технологиях разработки программного обеспечения, а не на фиксированных программных продуктах, их реализующих, набор используемого при обучении инструментального программного обеспечения может варьироваться в зависимости от со-

стояния рынка и возможностей реализующего данную методическую систему вуза.

Педагогический эксперимент по внедрению данной методической системы в процесс обучения студентов института информатики и информационных технологий Уральского государственного педагогического университета, позволил выявить применимость и целесообразность методической системы для обучения программированию студентов как технических, так и педагогических специальностей.

ЛИТЕРАТУРА

1. Alexeevskiy, P.I. Teaching Computer Programming to Students of Pedagogical Specialties through the Process of Game Development. / P.I.Alexeevskiy. // The Ethos of the Academe: Standing the Test of Time / Ariel University, Ariel, 2013

2. Jacobson, Ivar; Booch, Grady; Rumbaugh, James. The Unified Software Development Process, Addison Wesley Longman, Inc., January 1999, ISBN: 0-201-57169-2

3. Алексеевский П.И. Изучение средств управления версиями в рамках курса по программированию / П.И. Алексеевский // Инновационные технологии в образовательном процессе высшей школы / УрГПУ, Екатеринбург, 2012

4. Алексеевский П.И. Обучение программированию студентов на основе методологии унифицированного процесса разработки программного обеспечения / П.И. Алексеевский // Педагогическое образование в России, №8/2014 / УрГПУ, Екатеринбург, 2014

5. Зарукина Е.В. Активные методы обучения: рекомендации по разработке и применению: учеб.-метод. пособие / Е.В. Зарукина, Н.А. Логинова, М.М. Новик. СПб: СПбГИЭУ, 2010.

7. МОДЕЛИРОВАНИЕ СЛОЖНЫХ ОБЪЕКТОВ

В курсе компьютерного моделирования рассматриваются, в основном, задачи, для которых сформулировать математическую модель достаточно просто. В то же время, на практике приходится сталкиваться с так называемыми трудно формализуемыми, задачами, для которых построение математической модели связано с множеством условностей. В книге Самарского и Михайлова [10] этим задачам отведена целая глава. Но при рассмотрении задач о соперничестве рассматриваются только два противоборствующих субъекта. Если же добавить хотя бы ещё одного игрока на этом поле, то задачу в аналитической форме решить проблематично. То есть, возникают те же трудности, что и в классической

механике при решении задачи *n*-тел при *n* равном трём и более. Между тем, такие задачи очень часто возникают в физике, химии, биологии, а в стандартных курсах компьютерного моделирования они практически не рассматриваются.

Продемонстрируем два подхода к моделированию многочастичных задач на примере рассмотрения кристаллической структуры твердого тела. Первый подход связан с моделированием динамики дефектов кристаллической структуры. Второй – с использованием метода кинетики химических реакций для изучения дефектной структуры.

Кристаллы – это вещества, обладающие теми или иными симметриями. В настоящее время достаточно хорошо изучено порядка ста тысяч различного рода кристаллов, причем около двадцати тысяч из них органического происхождения. Кристалл обладает определенной структурой, которая может быть нарушена различного рода дефектами: примесями, вкраплениями, объединениями нескольких кристаллических веществ. Поэтому в теоретической кристаллографии структура кристалла представляется кристаллической решеткой. Например, на рисунке представлены элементарные ячейки двух кристаллических решеток: слева объемно-центрированная кубическая (ОЦК), справа гранецентрированная (ГЦК). Элементарной ячейкой этих решеток является куб со сторонами, равными постоянной решетки. ОЦК решетка присуща железу, натрию, хрому, вольфраму и т.д. ГЦК решеткой обладают благородные металлы: золото, серебро, платина, а также медь, никель, алюминий и др.

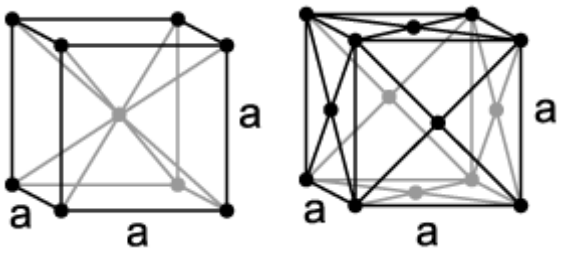

Рисунок 1. Элементарные ячейки ОЦК и ГЦК решеток

Идеальный кристалл представляет собой бесконечно большую совокупность элементарных ячеек, т.е. обладает трансляционной симметрией. Всякое нару-

шение трансляционной симметрии называется дефектом кристаллической решетки. Дефекты подразделяются на точечные, линейные (дислокации), двумерные (дефекты упаковки, границы зерен), трехмерные (поры, скопления примесей). Дефекты оказывают влияние на физические свойства материала (напр., прочностные характеристики). Поэтому исследованию дефектной структуры уделяется большое внимание.

7.1. **Метод молекулярной динамики**

Метод молекулярной динамики (ММД) или метод частиц широко используется для решения задач астрофизики, моделирования полупроводниковых приборов, моделирования твердого тела, жидкости, фазовых переходов [12]. Самые первые вычислительные эксперименты с системой твердых шаров были выполнены в 1959 году Олдером и Вэйнрайтом [13]. Однако по настоящему пионерской работой по разработке метода молекулярной динамики следует считать работу сотрудников Брукхейвенской лаборатории (США) [14], построивших компьютерную модель радиационного повреждения кристаллической решетки при нейтронном облучении. В этой модели удалось не только проследить движение каждого атома и развитие каскада столкновений выбиваемых атомов, но и открыть так называемый эффект каналирования первично выбитых атомов решетки (ПВА), подтвержденный впоследствии экспериментально сотрудником НИИЯФ МГУ А.Ф. Тулиновым.

Таким образом, уже в этой работе было показано, что компьютерное моделирование занимает промежуточное звено между теорией и экспериментом. Со стороны теории в вычислительном эксперименте служит математическая модель интересующего нас физического явления, а со стороны эксперимента – возможность изменять условия его проведения. Можно варьировать энергию частиц, вызывающих радиационное нарушение, угол вылета ПВА, температуру, концентрацию примесей и т.д. В модели частиц уравнения математической модели переводятся в дискретную алгебраическую форму, поддающуюся численному интегрированию.

Идея ММД чрезвычайно проста. Мы записываем уравнения движения классической механики для каждой частицы.

$$d^2 r_k^i / dt^2 = m^{-1} F_k^i(t), \qquad (1)$$

где $k = x,y,z;$, а $i = 1,2,\ldots, n$ (n – количество частиц, рассматриваемых в модели). Т.е. необходимо решить систему, насчитывающую $3n$ уравнений. Численное интегрирование системы (1) осуществляется методом конечных разностей, для чего ее следует записать в виде:

$$v_k^i\left(t + \frac{\Delta t}{2}\right) \cong v_k^i\left(t - \frac{\Delta t}{2}\right) + m^{-1}\Delta t F_k^i\{r_k^i(t), v_k^i(t)\} \qquad (2)$$

$$r_k^i(t + \Delta t) \cong r_k^i(t) + \Delta t v_k^i\left(t + \frac{\Delta t}{2}\right). \qquad (3)$$

В (2) и (3) Δt – временной шаг; F_k^i – k-я компонента силы взаимодействия i-го атома с остальными в пределах радиуса действия потенциала; r_k^i – k-я координата i-го атома; m – масса атома.

Задав начальные положения атомов и их скорости, с помощью (2) и (3) можно определить траектории их движения в последующие моменты времени, т.е. решить стандартную задачу Коши. Конечно, для реального кристалла такая задача невыполнима, поскольку количество атомов в нем порядка 10^{23}. Поэтому сначала необходимо построить физическую модель кристалла, насчитывающую всего несколько сотен или тысяч атомов: так называемый кристаллит, который сохраняет кристаллическую структуру изучаемого объекта. Чтобы имитировать целый кристалл, необходимо задать граничные условия.

Самый простой вид граничных условий: циклические граничные условия. В этом случае скорости частиц на границе кристаллита равны скоростям на противоположной границе с обратным знаком. Однако более реалистичны граничные условия, имитирующие переход к бесконечному упругому континууму. В этом случае на поверхностные атомы действую три вида сил. Постоянная сила, уравновешивающая отталкивание внутренних атомов, упругая сила, характеризующая реакцию внешнего окружения на смещение поверхностного

атома и вязкая сила, определяющая диссипацию энергии из рассматриваемой области кристаллита.

Преимущество компьютерного моделирования состоит в том, что сама модель является предметом исследования. Поэтому коэффициенты упругой и вязкой силы можно подобрать достаточно реалистичные, контролируя выполнение закона сохранения энергии (потенциальной энергии взаимодействия всех атомов плюс кинетическая энергия их движения).

Вторая проблема построения такой компьютерной модели – это выбор потенциала взаимодействия атомов друг с другом. Первые попытки сконструировать такой потенциал предпринимались еще в 20-х годах прошлого столетия[15,16]. С развитием зонной теории твердого тела в 60-70 годах предпринимались многочисленные попытки по расчету межатомного взаимодействия (теория псевдопотенциала) [11]. Однако эти усилия не увенчались особым успехом.

Во-первых, используемый в машинной модели потенциал должен быть ограничен областью взаимодействия только с ближайшим окружением, чтобы расчеты не были слишком громоздки. Во-вторых, на расстоянии радиуса обрезания потенциала он должен быть равен нулю, чтобы при расчетах не происходило скачков потенциальной энергии кристаллита. И, в-третьих, расчеты макроскопических характеристик исследуемого кристалла (объёмный модуль упругости, фононный спектр и т.д.) должны удовлетворять эксперименту. В своё время автором был построен такой потенциал, позволивший рассчитать микроскопические характеристики различного рода дефектов [8,9].

Третья проблема ММД – выбор шага интегрирования системы (2), (3). При слишком большом шаге Δt траектории выбиваемых из своих узлов атомов могут расходиться с появлением точек бифуркации. При слишком маленьком – помимо того, что увеличивается машинное время, в расчетах накапливается ошибка, связанная с приближенным вычислением на каждом шаге. Поэтому величину Δt лучше подбирать эмпирическим путем.

В заключение этого раздела отметим, что созданная в работе [14] модель позволила решить в последующие годы самые разнообразные задачи, связанные с возникновением и развитием радиационных нарушений, а также с атомной конфигурацией дефектной структуры. Довольно подробные обзоры этих работ были сделаны в журналах УФН [1,3]. Кроме радиационной физики твердого тела, компьютерное моделирование методом молекулярной динамики, как уже было отмечено, в настоящее время широко используется для исследования ионных кристаллов, полупроводников, жидкостей, фазовых переходов, химических соединений. ММД нашел также применение в биохимии и биофизике [17].

В работе Нормана и Стегайлова [7] дано стохастическое обоснование метода молекулярной динамики.

Появление новых конструкционных материалов ставит задачи более глубокого исследования влияния на них радиационного воздействия [4] и изучения дефектной структуры с помощью ММД [2].

Изучение физики наноструктур под действием облучения также заставляет прибегать к использованию компьютерного моделирования [5,6].

7.2.**Метод кинетики химических реакций**

Скорость образования нового вещества в результате протекания химической реакции двух реагентов можно представить в виде уравнения:

$$\frac{dC}{dt} = KC_1C_2, \qquad (4)$$

где C_1 и C_2 - концентрации реагентов, C – концентрация нового вещества, K – кинетический коэффициент, определяемый энергией активации реакции. Для того чтобы вступить в реакцию, частицам необходимо оказаться в непосредственной близости друг от друга. Вероятность такого события есть вероятность произведения событий попадания частиц в один и тот же объем и при этом об-

ладать энергией активации реакции. Таким образом, K определяется уравнением Аррениуса

$$K = K_0 \exp\left(-{}^{Q}/_{kT}\right),$$

(5)

где Q – энергия активации реакции, k – постоянная Больцмана, T – абсолютная температура.

Уравнение (4) описывает бимолекулярную реакцию, но приближение химической кинетики может быть распространено и на более сложные случаи, в том числе, на реакции взаимодействия дефектов в кристаллическом твердом теле. На рис.1 показаны элементарные ячейки для ОЦК и ГЦК решеток. Кристалл идеальной структуры есть повторение этих ячеек в трех направлениях. Но идеальный кристалл на самом деле некоторая иллюзия. Реальный кристалл имеет множество нарушений симметрии. Например, поверхность нарушает трансляционную симметрию. Идеальность кристаллической решетки нарушается также за счет различного рода дефектов. К таким дефектам относятся точечные дефекты и их скопления в виде линейных, плоских и объемных дефектов.

Чужеродный атом, замещающий атом в узле кристаллической решетки, называется примесью замещения. Узел, в котором отсутствует атом, называется вакансией. Собственный или чужеродный атом, находящийся в каком-либо положении, несовпадающем ни с одним из узлов решетки называется дефектом внедрения. Примесные дефекты неизбежно возникают в процессе кристаллизации или могут быть имплантированы с помощью ионных ускорителей. Собственные дефекты (вакансии и междоузлия) могут возникнуть как за счет тепловых колебаний атомов решетки, так и, например, за счет воздействия на кристалл радиационного излучения (например, нейтронами), механических напряжений и т.д.

Точечные дефекты вследствие тепловых колебаний атомов решетки могут диффундировать по кристаллу, вступать в реакции друг с другом, погло-

щаться стоками для дефектов, такие как поверхность, дислокации, границы зерен и т.д. Например, облучение кристалла высокоэнергетическими частицами приводит к выбиванию атомов из узлов решетки. При этом образуются так называемые френкелевские пары: вакансия + междоузельный атом (МУА). МУА обладает высокой подвижностью даже при сравнительно низких температурах. Перемещаясь, они могут аннигилировать с собственной или с другой вакансией, взаимодействовать друг с другом или с атомами примеси, образуя объемные неподвижные дефекты. Изучение этих процессов необходимо для понимания изменения свойств конструкционных материалов.

Рассмотрим ситуацию, когда в кристалле, содержащем атомы примеси, образуются за счет облучения вакансии и МУА. При этом температура такова, что вакансии и примеси неподвижны. Тогда для изменения концентраций дефектов можно записать уравнения, весьма похожие на уравнения Лотки-Вольтерры, если под хищниками понимать вакансии и атомы примеси.

$$dC_i/dt = g - K(r_t C_t) + r_v C_v) C_i$$

$$dC_v/dt = g - K r_v C_v C_i , \quad (6)$$

где - концентрации МУА, вакансий и атомов примесей, являющихся ловушками для МУА; K – кинетический коэффициент, пропорциональный коэффициенту диффузии для МУА, т.е. определяющий вероятность взаимодействия МУА с вакансией или атомом примеси;

радиус захвата МУА с вакансиями и атомами примеси, соответственно; g - скорость рождения френкелевских пар.

Поскольку каждая френкелевская пара приводит к изменению электросопротивления изучаемого образца, то общее изменение электросопротивления будет пропорционально концентрации образующихся вакансий. Поэтому для заданной концентрации примесей путем сравнения изменения электросопротивления с результатами численного решения системы уравнений (6) можно определить энергию активации МУА, а также радиусы его взаимодействия с примесными атомами.

ЛИТЕРАТУРА

1. Агранович В.М., Кирсанов В.В. Проблемы моделирования радиационных повреждений в кристаллах // УФН, 1976, Т. 118, вып.1, с.3-51.
2. Карькина Л.Е., Яковенкова Л.И. Моделирование атомной структуры дефектов в кристаллах: научно-образовательная серия «Физика конденсированных сред», Екатеринбург: УрО РАН, 2011. - 463 с.
3. Кирсанов В.В., Орлов А.Н. Моделирование на ЭВМ атомных конфигураций дефектов в металлах//УФН, 1984, Т. 142, вып.2, с.221-264.
4. Козлов А.В. // Действие нейтронного облучения на металлы при различных температурах и возможность самоорганизации протекающих при этом процессов // Физика элементарных частиц и атомного ядра, 2006, Т.37, вып.4, с.1109-1150.
5. Лагунов В.А., Синани А.Б. //Компьютерное моделирование межузельных атомов в двумерных нанокристаллах // ФТТ, 2003, Т. 45, вып.3, с.542-547.
6. Новиков Л.С., Воронина Е.Н. // Особенности моделирования радиационных воздействий на наноструктуры: Труды XIII Межвузовской научной школы молодых специалистов "Концентрированные потоки энергии в космической технике, электронике, экологии и медицине", Москва 19-20 ноября 2012, с.133-141
7. Норман Г.Э., Стегайлов В.В. //Стохастическая теория метода классической молекулярной динамики// Матем. моделирование, 2012, Т.24, вып.6, с.3–44

8. Подчиненов И.Е., Плишкин Ю.М.//Парный потенциал взаимодействия атомов в меди, используемый для расчета характеристик дефектов//ФММ, 1973, Т. 36, вып.2, с.260-264.
9. Подчиненов И.Е.//Парный потенциал взаимодействия, используемый в машинных моделях: аналит. Обзор, 1975 / Академия наук СССР, Институт физики металлов УНЦ. М.: ВИНИТИ, №1306-75.
10. Самарский А.А., Михайлов А.П. Математическое моделирование: Идеи. Методы. Примеры: 2-е изд., испр: М.: Физматлит, 2001. – 320 с.
11. Харрисон У. Псевдопотенциалы в теории металлов. М.: Мир, 1968. – 367 с.
12. Хокни Р., Иствуд Дж. Численное моделирование методом частиц. М.: Мир, 1987. - 638 с.
13. Alder B.J.,Wainwright T.E. Studies in Molecular Dynamics. I. General Method. - J.Chem.Phys., 1959, vol. 31, pp. 459-466.
14. Gibson I.B., GolandA.N., Milgram M.M. and Vineyard G.H. Dynamics of Radiation Damage. – Phys.Rev., 1960, vol. 120, pp.1229-1253.
15. Jones J.E. // On the Determination of Molecular Fields. II. From the Equation of State of a Gas. – Proc.R.Soc., 1924, A106, pp.463-477.
16. Morse P.M. // Diatomic Molecules According to the Wave Mechanics. II. Vibrational Levels. – Phys. Rev., 1929, vol. 34, pp.57-64.
17. Sbalzarini Ivo F.: Analysis, Modeling & Simulation of Diffusion Processes in Cell Biology: VDM Publishing, 2009, 388 p.

Инновационные технологии в подготовке учителей информатики

Подписано к печати 12.04.2016 Формат 60х84 1/16

Бумага для множ. ап. Гарнитура «Таймс» Печать на ризографе.

Усл. печ. л. 7,5. Тираж 100 экз. Заказ №

Редакционно-издательский отдел ФГБОУ ВО «Уральский государственный

Педагогический университет»

Отдел множительной техники

620017, г. Екатеринбург, пр. Космонавтов, 26

e-mail: uspu@uspu.ru